Moving towards software-defined security in the era of NFV and SDN

Montida Pattaranantakul

université PARIS-SACLAY

NNT : 2019SACLL009

Thèse de doctorat

Moving Towards Software-Defined Security in the Era of NFV and SDN

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom SudParis

École doctorale n°580 Sciences et Technologies de l'information et de la Communication (STIC)
Spécialité de doctorat: Réseaux, Information et Communications

Thèse présentée et soutenue à IMT Lille Douai, Villeneuve d'Ascq, le 20 Juin 2019, par

Montida PATTARANANTAKUL

Composition du Jury :

M. Frédéric CUPPENS Professeur -- IMT Atlantique	Rapporteur
M. Youki KADOBAYASHI Professeur -- NAIST, Japan	Rapporteur
Mme. Nora CUPPENS Directrice de recherche -- IMT Atlantique	Examinatrice
M. Michaël HAUSPIE MdC (HDR) -- University of Lille	Examinateur
Mme. Houda LABIOD Professeure -- Télécom ParisTech	Examinatrice
Mme. Maryline LAURENT Professeure -- Télécom SudParis	Examinatrice (Président du jury)
M. Zonghua ZHANG Professeur -- IMT Lille Douai	Directeur de thèse
M. Ahmed MEDDAHI Professeur -- IMT Lille Douai	Co-Directeur de thèse

To my beloved family…

Acknowledgments

The completion of this thesis would have not been possible without the help, support, advice and guidance I have received from many people in my graduate life. Some of them have unexpectedly come to our live to give lessons, while others intentionally come to give us blessing, inspiring, encouraging, or even sharing some ideas on how to deal with the problems more calmly and reduce the stress in everyday life. I would like to extend my sincere appreciation to all of them from very deep inside.

First and foremost, I would like to express my gratitude to my supervisor, Prof. Zonghua Zhang for giving me the great opportunity and leading me to the hall of Network Functions Virtualization (NFV) and Software-Defined Networking (SDN). His guidance, enthusiasm, immense knowledge, continuous support, motivation and critical feedbacks throughout the years have been great contributions to my thesis and professional development. It is absolutely difficult to have the good quality of publications and succeed in the process of developing significant ideas without his help. His passion for research and rigorous research approaches have influenced me immensely.

I am also extremely thankful to my co-supervisor, Prof. Ahmed Meddahi for his continuous guidance and support during my thesis. Thank you so much for your time and dedication, your remarks always appropriate and direct to the point. Also, thank you so much for your help and advices at every stages of my thesis for both administrative and research works. I am grateful for getting the opportunity to work with him. I am also extremely thankful to Prof. Abdallah M'Hamed from Télécom SudParis to help me contact administrative staffs and accomplish the administrative works at the early stage of my Ph.D registration. Thank you so much for your promptly response and fully support whenever I need help.

I would also like to thank Prof. Frédéric Cuppens and Prof. Youki Kadobayashi for agreeing to serve as the reviewers of my thesis and provide insightful comments. My particular gratitude goes also to the jury members, Prof. Nora Cuppens, Prof. Michaël Hauspie, Prof. Houda Labiod, and Prof. Maryline Laurent, for their acceptance to be as a part of the thesis committee and evaluate my work.

Special thanks to Dr. Ruan He who is considered as the key driver of NFV project, providing research grant and giving me the guidance on developing security orchestrator. His dedication and professionalism have inspired me. Also, his valuable experiences and feedback sharing from industry point of view, leading me to gain a holistic understanding on NFV/SDN industry research road map and grasping that security in NFV raises important concerns.

I would like to thank Dr. Qipeng Song and Dr. Yuchia Tseng for your time and effort you have made in helping me for research works throughout the formal and informal discussions, brainstorming, or even step-by-step guidance about developing and implementing the idea.

I would like to thank all IMT Lille and SAMOVAR lab's staffs for handling all the issues related to my resident permit, attestation, missions, *etc.*, during my study. It would definitely be difficult to complete the tasks without your kind support.

I am indebted to many kindhearted people outside the school, who have supported me in one way or the other during this amazing journey. First, I extend my deepest gratitude to my research institute, National Electronics and Computer Technology Center (NECTEC), a member of National Science and Technology Development Agency (NSTDA), Thailand, that offers a full scholarship for me to pursuing master and doctoral degrees in France. Second, I would like to extend my heartfelt thank to Dr. Saran Sumriddetchkajorn (Executive director of NECTEC during 2014 - 2018), and Dr. Chalee Vorakulpipat (Head of Cybersecurity laboratory, NECTEC) for the great opportunity you have given in order to gain in-depth knowledge, skills and research abilities on the new networking paradigms like NFV and SDN. Thank you so much for your regularly following up my research progress and advise me in advance. Third, I would like to thank Mme. Panida Rojrattanachai (Minister Counsellor of Education, France), Mme. Nareenush Kaopaibool (Education senior officer, France), and M. Somchai Injorhor (Education senior officer, NSTDA, Thailand) for overwhelming support over the years, such as assisting all services related to local law and regulation, providing some guidance for health care, resident permit, accommodation, reimbursement, personal issues, *etc.*. Thank you for making my stay in France less stressful. Forth, my sincere thanks also goes to my friends, Weerapan Sae-dan, Paul Tamoyo-Serrano, Napeesah Bing, Soepuroe Hayeechepute, Amran Vani, Siriporn Saiburee and other NECTEC friends, who are more like brother for me. Thank you very much for listening my problems, supporting and encouraging me all the time I needed. You are truly my blessing.

I would like to thank my dearest family with all my heart. I could not imagine life without your caring and supportive love. You are the reason for me to be thankful and feel blessed every time. Words are not enough to express my deep and sincere gratitude to my parents. Thank you so much for all care and love, while waking up at the late night and praying for me in the removal of all obstacles and success in all noble endeavors. A massive thank you to my beloved grandmother and grandfather, Hajah Sarome and Al-Marhum Wan-Dani Wantalabe for teaching me the value of hard work and perseverance. I am so blessed that you cultivated this *never give up* attitude in me, which has continued to help me success in life. I also extend my huge gratitude to all other family members. Lastly, I would like to give a thank to my boyfriend Attasit Poomkleang and his family for caring about me, and always keep me in their prayers. A special thanks to my boyfriend for always stay beside me whatever I feel happy or sad. I cannot thank you enough for being there all the time, encouraging me to follow my dream and supporting every step of the way.

Abstract

Network Functions Virtualization (NFV), along with Software Defined Networking (SDN), drives a new change in networking infrastructure with respect to designing, deploying, and managing various network services. Both of them essentially rely on software based approaches, while operating at different levels of the network. In particular, NFV has potential to significantly reduce the hardware cost, greatly improve operational efficiency, and dramatically shorten the development lifecycle of network services. It also makes network services and functions much more adaptive and scalable to meet the rapid growth and change of user requirements. As a two-sided coin, despite these well-recognized advantages, security remains to be one of the vital concerns and potential hurdle. On one hand, it is unclear how NFV and SDN would fundamentally impact the landscape of cyber defense, and how it can help to improve security management. On the other hand, novel security threats and vulnerabilities will be inevitably introduced, potentially resulting in broader attack surface and blurred defense lines in the virtualization environment.

This thesis is intended to explore security issues in the virtualized and software-defined world, and starts with two important hypotheses: (1) SDN and NFV offer plenty of opportunities for us to rethink security management in the new networking paradigms; (2) both legacy and new security threats and vulnerabilities in NFV/SDN enabled environments need to be sufficiently addressed in order to pave the way for their further development and deployment. To validate the hypotheses, we carry out an in-depth study on NFV/SDN from security perspective, including its architecture, management and orchestration (MANO) framework, and use cases, leading to two major contributions, (1) a security management and orchestration framework (called SecMANO) based on NFV MANO, which has the potential to manage a set of policy-driven security mechanisms, such as access control, IDS/IPS, network isolation, data protection; (2) a comprehensive threat analysis on five NFV use cases and the state-of-the-art security countermeasures, resulting in a NFV layer-specific threat taxonomy and a set of security recommendations on securing NFV based services.

We believe that both of the two contributions lay down a foundation for security research in NFV/SDN domain. In particular, based on the two contributions, we further develop a security orchestrator as an extension of available NFV orchestrator, with an objective to enabling the basic security functions to be effectively orchestrated and provided as on-demand services to the customers, meanwhile allowing high-level security policies to be specified and enforced in a dynamic and flexible way. Specifically, a software-defined access control paradigm is implemented and prototyped with Tacker (an OpenStack service for NFV orchestration using TOSCA model), which allows the security administrators to dynamically customize the access control models and policies for different tenant domains, eventually achieving flexible and scalable protection across different layers and multiple cloud data centers. Both prototype of concept and real-life experiments on testbed have been carried out, clearly demonstrating the feasibility and effectiveness of our security orchestrator.

In addition, as our NFV cross-layer threat taxonomy indicates, a large set of novel threats will be introduced, among which VNF (Virtualized Network Function) is a unique and important asset that deserves careful protection. The fourth contribution of this thesis is therefore devoted to achieving secure and dependable SFC (Service Function Chaining) in NFV and SDN environment. Specifically, an identity-based ordered multisignature scheme called SecSFC is designed and applied to ensure that, (1) each service function involved in a particular service chain is authenticated and legitimate; (2) all the service functions are chained in a consistent, optimal, and reliable way, meeting with the pre-defined high-level policy specifications like VNF Forwarding Graph. Both theoretical security analysis and experimental results demonstrate that our scheme can effectively defend against a large set of destructive attacks like rule modification and topology tempering, moving an important step towards secure and dependable SFC. Importantly, the signature construction and validation process is lightweight, generating compact and constant-size keys and signatures, thereby only incurring minimal computational overhead and latency.

Résumé

Les réseaux programmables (SDN) associés à la virtualisation des fonctions réseaux (NFV), conduisent à de nouveaux paradigmes réseaux, en particulier dans le domaine de la conception, du déploiement et de la gestion de services réseaux. Les réseaux SDN et NFV s'appuient essentiellement sur des technologies orientées logicielles et opèrent sur différents niveaux ou couches réseaux. La virtualisation réseaux, permet de réduire de manière significative les coûts liés au matériel, d'améliorer les opérations de maintenance, mais aussi de réduire le cycle de développement et de déploiement des services réseau. Le caractère dynamique des fonctions réseaux, offert par la virtualisation, conduit à une meilleure adaptation et évolution des services réseaux déployés et par conséquent, une plus grande réactivité de la part des opérateurs, pour répondre à la croissance et à l'évolution rapide des besoins des utilisateurs. Les réseaux SDN/NFV contribuent ainsi à une plus grande automatisation du réseau. Néanmoins, dans les environnements SDN, NFV, l'aspect sécurité constitue un défi et un enjeu majeurs, qu'il est primordial de prendre en compte et d'étudier à tous les niveaux, de la conception au déploiement à grande échelle de ces réseaux. Dans ce contexte, il est nécessaire de mesurer l'impact des technologies NFV et SDN, sur la sécurité, dans le paysage des menaces cyber, mais aussi leur capacité à garantir et gérer la sécurité, dans des environnements essentiellement logiciels. De plus, dans ces environnements virtualisés, de nouveaux types de menaces et vulnérabilités sont inévitablement introduites, ce qui conduit potentiellement à une surface d'attaque plus large et à des contre-mesures associées plus délicates à mettre en œuvre.

Ce travail de thèse, vise à étudier les vulnérabilités en termes de sécurité dans les environnements réseaux logiciels et virtualisés, pour proposer des modèles et des mécanismes spécifiques, afin d'améliorer la sécurité dans ces environnements critiques. Nous considérons les deux hypothèses suivantes: (1) Les changements de paradigmes introduits par les réseaux SDN et NFV permettent de développer de nouvelles approches en matière de gestion de la sécurité; (2) L'ensemble des menaces et vulnérabilités dans les environnements NFV/SDN doivent être intégralement pris en compte, en particulier pour un développement et déploiement à grande échelle. Pour cela, dans une première partie, nous proposons une étude détaillée et complète, du point de vue de la sécurité, des architectures et protocoles SDN/NFV, mais aussi de la gestion et de l'orchestration des fonctions réseaux dans ces environnements (architecture MANO). Plusieurs cas d'usage ou scénarios sont spécifiés et proposés, en guise d'illustrations. Cette première étude a conduit à deux contributions majeures: (1) une architecture complète pour la gestion et l'orchestration de la sécurité (appelé SecMANO) basé sur NFV MANO. SecMANO permet de gérer un ensemble de fonctions service, de mécanismes de sécurité (contrôle d'accès, IDS/IPS, isolation, protection) basées sur un ensemble de règles; (2) une analyse complète des menaces et vulnérabilités dans le contexte NFV, à partir de cinq cas d'usage spécifiques, et des contre-mesures associées. Cette analyse a permis de proposer, une classification (taxonomie) complète et détaillée, des différents types de menace spécifique, associés à un ensemble de recommandations, pour une meilleure sécurité des services NFV.

Nous estimons que ces deux premières contributions ouvrent des perspectives de recherche intéressantes, dans le domaine de la sécurité des réseaux NFV/ SDN.

Cette première étude, nous a amenés à proposer en guise de troisième contribution, une nouvelle architecture pour l'orchestration de fonctions de sécurité dans les environnements virtualisés. Cet orchestrateur de sécurité a été spécifié et développé comme un module d'extension pour les orchestrateurs existants. L'objectif est d'assurer un déploiement dynamique, flexible, à la demande, ainsi qu'une orchestration efficace des différents services de sécurité de base. De plus, l'architecture prend en compte et applique de manière dynamique, la stratégie de sécurité, spécifiée par l'utilisateur à un haut niveau (politique de sécurité) à travers un modèle de données spécifique. Plus précisément, un mécanisme de contrôle d'accès, défini et appliqué à partir d'un langage de haut niveau, basé sur les piles "Tacker" (un service OpenStack pour orchestrateur NFV utilisant le modèle de donnés TOSCA), a été prototypé, implanté et testé. Ce prototype, permet de personnaliser et d'adapter dynamiquement, le modèle et la stratégie de contrôle d'accès, pour différents domaines utilisateurs concurrents. Ces domaines de sécurité indépendants, restent potentiellement protégés et isolés, dans les environnements à grande échelle, multi-opérateurs et multi-clouds. Le prototype et les expérimentations menées dans des conditions pratiques, montrent la faisabilité et l'efficacité de l'approche proposée.

L'étude et la classification proposées dans la première partie, à partir d'une approche "cross-layer", mettent en évidence de nouveaux types de menaces et vulnérabilités et démontrent que dans ces environnements logiciels, virtualisés, la sécurité est l'élément critique. La quatrième contribution (SecureSFC ou SecSFC) vise à sécuriser et à fiabiliser, la composition et le chaînage de fonctions service (Service Function Chaining – SFC) dans les environnements NFV/SDN. SecureSFC s'appuie sur un mécanisme de type "identity-based ordered multisignature" pour garantir les propriétés suivantes: (1) L'authentification de chaque fonction service, associée à une chaîne de fonctions service particulière; (2) La cohérence et le séquencement de l'ensemble des fonctions service associées à une composition ou à un chaînage particulier de fonctions service ("VNF forwarding graph"). L'analyse théorique du modèle proposé "SecSFC" et les résultats expérimentaux, montrent le caractère résilient de l'approche, en particulier face à un certain nombre d'attaques spécifiques (ex. modification des règles ou de la topologie) avec un temps de traitement et une latence, limités. Dans le domaine de la sécurité SFC, "SecSFC" constitue une étape majeure.

Contents

List of Figures

List of Tables

1 Introduction

Over the past decades, enterprise networks have undergone increasing diversity of network services and functions. The tremendous growth in vertical and horizontal deployment of proprietary network appliances makes network architectures extremely complicated and difficult to manage [213, 32]. In particular, typical network appliances always incur high costs of investment (Capital Expenditure (CapEx)) and maintenance (Operation Expenditure (OpEx)) [210], as they are usually expensive, vendor specific, and complicated to manage, creating numerous pain points to network administrators [88, 207]. Also, a new form of typical network appliance is usually treated as one-off solution to support a specific function [12], making it hard to deploy new service or customize the existing network appliances for fulfilling different customer requirements. An example in the deployment of today's network services, such as firewall, Intrusion Detection and Prevention System (IDS/IPS), is that the network functions are often hardware-dedicated and placed at fixed network spots, and the capacity of handling traffic is limited. For example, traditional hardware-based firewall fails to support dynamic provisioning to deal with frequent and highly variation of traffic load [58]. Also, it is a difficult and complex task to monitor the malicious processes of VMs running inside the host by using traditional IDS/IPS, since the traditional IDS/IPS only provides fixed functionality to deal with specific types of attacks, and has fixed capacity in handling network traffic [82].

To address the aforementioned issues, many enterprises and service providers are seeking for more effective techniques to improve operational efficiency, reduce power usages, and speed up their service deployments. Network Function Virtualization (NFV), along with Software-Defined Networking (SDN), has emerged as promising solutions in recent years. In particular, the concept of NFV was firstly proposed by the European Telecommunication Standard (ETSI) with the purpose to reduce hardware investment cost, enhance capacity of resource utilization, and accelerate service deployment of new network services to support business revenue and future growth objectives. The key idea of NFV is to decouple network functions from dedicated hardware devices and implement them using software-based approaches. That says, instead of installing, configuring, and operating a dedicated appliance to perform a network functions, NFV allows network operators to use standard hardware platform to simply load the software image into a virtual machine, and launch the desired network service on demand. As such, the network functions can be implemented and deployed on a range of commodity hardware located at different geographical locations, avoiding the needs of installing new equipments. Thanks to these characteristics, NFV-based implementations can significantly reduce capital and operational expenditures (CapEx and OpEx), increase network efficiency and agility, provide a shorter time to market deployment of network services, and improve the scalability of resource utilization [117, 98, 191]. Another silent feature of NFV is that a regular service function (*a.k.a*, Virtual Network Function (VNF)) can be broken down and decomposed into

smaller functional modules to accelerate the processing time, improving a better scalability and reusability (*e.g.*, [246, 58]). More interestingly, a diverse set of VNF instances can be chained together, so-called Service Function Chaining (SFC), to create on-demand network services in accordance with the particular needs of the users specification (*e.g.*, [186, 142, 146]).

In particular, NFV and SDN are two closely related technologies that use the network abstraction model but operate at different layers of NFV architectural framework. NFV aims to virtualize all network functions from the hardware on which it runs, allowing the network to grow without the addition of more devices. While SDN aims to separate control plane from forwarding plane, and moves decision making to the control plane. For example, it decides how to handle traffic at the data plane. This make it easier to configure, program and manage the networks. In other words, SDN is considered highly complementary to NFV, which is practically used as a part of SFC creation to provide the full network control capabilities and manage traffic steering at the data plane. They often exist together.

A typical use case illustrating the usage of NFV and SDN is service function chaining (SFC) or VNF chaining[1], as shown in the lower part of Figure 1.1. Specifically, a set of VNF instances are created, which are then stitched together in a logical ordered-fashion for creating a service chain or an end-to-end network service. In this use case, two essential functional blocks play the major role.

- **management and orchestration module** (*a.k.a*, NFV MANO [74]) of NFV, which maintains the full lifecycle management of infrastructure resources, VNFs, and network services. This include service initiation, configuration, update and termination, resource and service orchestration, scaling in/out, policy management and performance measurement. To date, many NFV MANO frameworks have become available, most of which are built according to ETSI NFV MANO specification. For example, CloudNFV [48] and CloudBand [6] are developed as non-model driven based approach, while the others have been developed under open-source platforms as model-driven NFV orchestration by using Topology and Orchestration Specification for Cloud Applications (TOSCA) standard [227], e.g., OpenStack Tacker [219], Cloudify [47], and ONAP [163].

- **SDN controller** [70, 165], which can be leveraged by NFV orchestrator to provide full network control capabilities, including traffic steering and forwarding, routing decision based on the global view of network status, and path provisioning between the instantiated VNFs. By leveraging such capabilities, a network operator can reconstruct a service chain and steer the traffic in a flexible and automated way. The available SDN controllers can be classified into two categories: (1) logically centralized controllers that maintain a global view of the network using a single controller, such as Floodlight [19], OpenDaylight [225], NOX [94], POX [183], RYU [159]; (2) Distributed controllers like ONOS [224], which aims to mitigate the limitations of the centralized controllers by deploying and synchronizing multiple controllers to share the communication and computation load.

1.1 Background and Challenges

Despite the promising advantages of NFV and SDN, security concerns remain to be significant barrier to their wide-spread adoption [179, 128, 84]. As new networking technologies, they are essentially two-sided sword from security perspective. On the one hand, novel security threats and vulnerabilities will be inevitably introduced. The resulting attack surface for NFV based

[1] In this thesis, VNF and Service Function (SF) are used interchangeably. VNF is defined by ETSI, while SF is specified by Internet Engineering Task Force (IETF).

network services could be even larger than its traditional counterpart, spanning from hardware vulnerabilities to the vulnerability of Virtual Network Functions (VNFs), orchestration, or even policy violation due to non-trivial service complexities and administrative errors. Especially, when the large scale deployment of NFV goes across a wide range of cloud data-centers and security domains, the frequent migration can bring a large set of challenges to threat landscape identification and security policies enforcement. On the other hand, the novel features and capabilities of NFV and SDN provide unprecedented opportunities to reshape the landscape of today's cyberdefense, potentially making significant progress towards next generation security architecture. For instance, the agility of NFV allows different security functions to be implemented as VNFs and deployed on demand. Also, the programmability of SDN may enable security functions to be implemented as network apps and customized via SDN controllers. To date, we have seen tremendous efforts paid to security research in both NFV and SDN, and two representative surveys are [206, 179]. However, in this thesis, we generally treat SDN as part of NFV, so SDN will not be explicitly mentioned unless it's particularly needed.

As a matter of fact, a majority of NFV MANO frameworks (*e.g.*, [219, 47, 223, 238, 6, 48]) is focused on the migration of network functions from dedicated hardware appliances to virtualization environment, as well as the effective management and orchestration of the virtualized network services. For example, the way to allocate virtualized resources, deploy and configure VNF instances, and create a service chain for delivering end-to-end network service. Although NFV offer many potential benefits, it is still at an early stage of development and deployment, security and regulatory concerns are comparatively overlooked. In particular, it remains unclear how NFV would fundamentally impact the landscape of cyber defense, how it can help to improve overall security posture, or even what kinds of security challenges as well as critical threats they may presented. For example, the authors of [116, 7] have raised some security concerns and proposed a security orchestrator to manage security mechanisms, while the detailed data models and the related use cases were not sufficiently addressed. Therefore, it is interesting and important to develop a novel security orchestration that can be seamlessly integrated with the existing NFV orchestrators, enabling the basic security functions to be orchestrated and provided as on-demand services to the customer, and high-level security policies to be specified and enforced in a dynamic and flexible way. In doing so, *security-by-design* and *security-as-a-service* have potential to be mostly, if not completely, achieved.

By design, NFV-enabled service deployment relies on multiple control layers, such as infrastructure layer which allocates physical and virtual resources, VNF layer which implements, deploys and manages the virtualization of network functions, and orchestration layer which manages and orchestrates network elements and services. Clearly, the open layered structure opens a door to a large set of novel security threats and vulnerabilities, significantly enlarging the attack surface of NFV. For instance, those attacks occurring at VMs (*e.g.*, [128, 181, 217, 109]) still exist and play a key role in triggering the so-called *cross-layer attacks*, which happen at different NFV layers. In order to help the service providers and network operators to gain a holistic understanding on the attack surface when implementing or deploying their network services in NFV environment, so as to deploy cost-effective security hardening based on their particular needs, it is very meaningful and important to establish a comprehensive threat taxonomy to systematically identify the threats and vulnerabilities in NFV.

In addition, as a unique feature of NFV and SDN, service function chaining (SFC) plays a key role in implementing and delivering end-to-end network services, which involves several steps from defining high-level network service description to resource allocation, VNF instantiation and placement, to VNF selection and composition, to traffic steering and forwarding. According to NFV reference architectural framework, NFV MANO stack lays a foundation to maintain the entire lifecycle of VNFs and network services, and SDN controller is mainly in charge of management of network connectivity and traffic flows in data plane. Although

there are several ongoing research work attempting to address the unique and unprecedented challenges imposed by SFC, we observe that the security and dependability problems in SFC are comparatively overlooked by the research community. One challenging issue, for example, is that once the high-level SFC policy is specified, how can we make sure that a service chain specification is correctly translated into network flow classification with accurate packet forwarding rules, or how to ensure and prove that the packet flows of a particular service chain are traversed correctly through all appropriate and legitimate VNFs in the right order with respect to the predefined policy. Anomalous flow redirection and path deviation [205, 247] are potential attack models that can be used by attackers to compromise the original service function path and further violate SFC policies. As a result, the attackers can bypass or evade from security functions in the service chain like firewall, IDS/IPS. Therefore, it is important to achieve both security (e.g., authenticity, integrity of VNFs) and dependability (e.g., ordering property), in addition to optimality, of the service chain in NFV environment.

1.2 Contributions

To address the aforementioned challenges, this thesis delivers the following contributions,

- A conceptual design framework about NFV based Security Management and Orchestration (called SecMANO), which aims to facilitate the basic security functions to be automatically and intelligently deployed, so as to protect the NFV assets according to the predefined security policies. The design of such a framework has two purposes: implementing the concept of *security by design* by formally specifying the security attributes of interest at the early stage of deployment; achieving *security as a service* by providing a basic set of security functions or their combinations on demand, *e.g.*, access control, IDS/IPS, network isolation, data protection. The existing NFV orchestration platforms have been studied from security perspective, with an objective to understanding their capabilities of managing security mechanisms, which finally lead to a comparative analysis on the existing NFV orchestration platforms in terms of the criteria for developing software-defined security framework.

- An in-depth analysis about NFV security. Starting with the analysis on the well-defined NFV uses cases documented by ETSI, we examined their potential threats and vulnerabilities, and established a comprehensive NFV layer-specific threat taxonomy, finally provided a suite of security recommendations based on the gap analysis with the available countermeasures.

- A security orchestrator, which can be treated as an implementation of the proposed conceptual framework SecMANO. Specifically, the security orchestrator is expected to work as an extension of the existing NFV orchestrator, which can deploy the basic security functions in NFV environments and optimally customize and orchestrate them as on-demand services to the customers, according to the high-level security policies. In particular, a software-defined access control paradigm has been developed, which allows access control models and policies to be specified and enforced in a dynamic and flexible way, providing tenant-specific cross-layer protection in the distributed clouds.

- A lightweight authentication scheme that achieves secure and dependable VNF chaining. This security scheme can be implemented and naturally integrated into SecMANO by interacting with VNF forwarding graph module and SDN controller. The design foundation of this scheme is aggregated ordering digital signature crypto primitive, which allows a

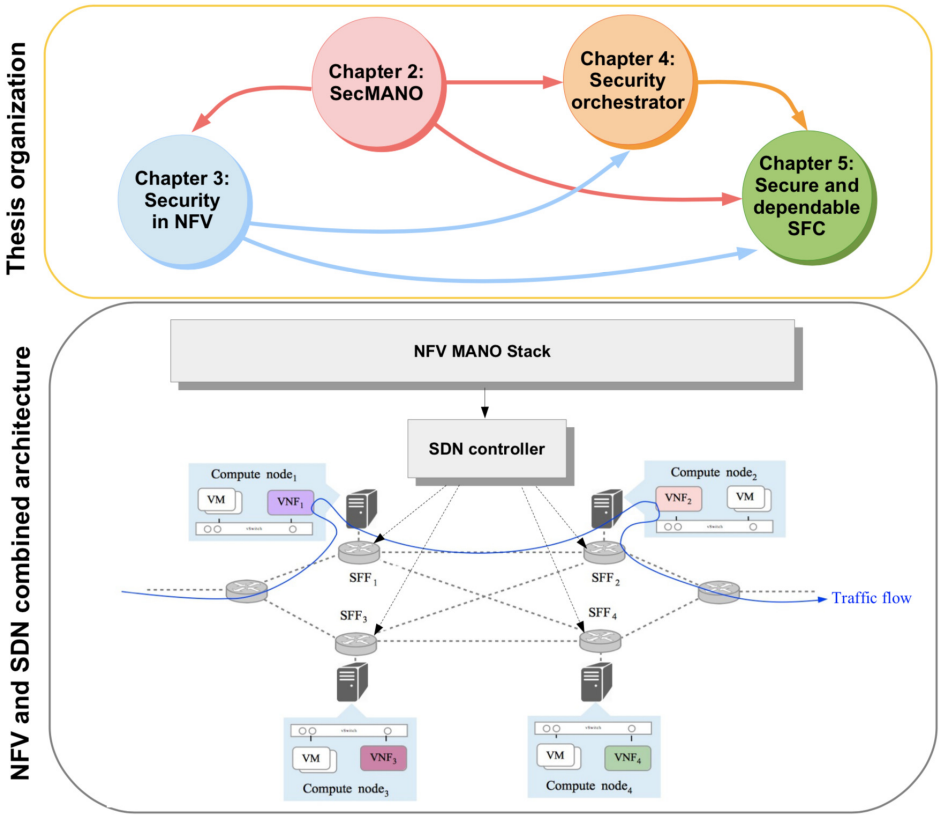


Figure 1.1: Organization of the thesis

group of VNFs involved in a particular service chain to attest their signatures on the packets received, and a verifier to later verify that all the packets traversed correctly w.r.t the specified SFC policy.

1.3 Outline of dissertation

The organization of this dissertation is given as follows, and the relationship between the sections is illustrated in Figure 1.1.

- **Chapter 2 – Security Management and Orchestration in NFV.** This chapter is dedicated to the first contribution about conceptual framework SecMANO. It firstly analyzes the issues in the traditional networking model, and highlights the significant advantages that can be achieved by NFV and SDN. Then NFV architectural framework and its relationship with SDN are discussed. Also, the existing frameworks related to NFV MANO are intensively studied in order to analyze their architectures, data models, and features. Our study reveals that the existing frameworks lack a dedicate module or component that can provide holistic security management. Therefore, a conceptual design framework of Security management and Orchestration (SecMANO) is proposed to help improve security management in NFV environment.
- **Chapter 3 – Security in NFV.** This chapter is focused on the second contribution, and its aim is three-fold. First, we conduct a comprehensive study on several NFV-based use cases that are documented by ETSI, and each of them is analyzed with their specific threats and vulnerabilities, with an ultimate goal to establishing a NFV layer-specific threat taxonomy. Such a taxonomy allows service providers and network operators to gain

a holistic understanding on the attack surface introduced by NFV. Second, we conduct a set of comparative studies on the typical security mechanisms (*e.g.*, Identity and Access Management (IAM), IDS/IPS, network isolation, and data protection) between their implementations and deployments in traditional scenarios and in NFV environments. Third, based on the established threat taxonomy and the comparative studies of security mechanism, we give a set of recommendations on securing different NFV layers. This may help network operators to understand to what extent the available security best practices for NFV can fulfill the security requirements.

- **Chapter 4 – A security orchestrator and its implementation for achieving software-defined access control.** This chapter specifically reports the third contribution, *i.e.*, security orchestrator that can be integrated seamlessly with the existing NFV orchestrators. One of the major motivations is to improve agility and flexibility of security functions, providing autonomic multi-layered defense mechanisms and fine-grained security control, while allowing security administrators to freely configure security services based on their particular contexts and needs. A software-defined access control paradigm is developed to illustrate the usage of our proposed security orchestrator. With such a novel access control paradigm, different tenants of the cloud can specify their own access control models and polices, and update them on the fly.
- **Chapter 5 – Towards Secure and Dependable Service Function Chaining (SFC).** This chapter reports the fourth contribution, a scheme dealing with the consistency and reliability challenges in SFC. Specifically, a new cryptography primitive, called *Lite identity-based ordered multisignature scheme* is proposed to examine the behavior of packet traversal and verify the consistency of service chain. The scheme impels each VNF instance in the specified service chain to attest its signature in the packets received. While, the egress node (SFC boundary node that handles traffic leaving the SFC-enabled domain) plays a role as verifier to examine the consistency of service chain by verifying the aggregated signatures, so as to validate the authenticity of the VNFs and their ordering properties (positions) in the service chain.
- **Chapter 6 – Conclusion.** This chapter concludes the dissertation with a summary of contributions and presents the perspective for future work.

2 Security Management and Orchestration in NFV

In this Chapter, we firstly address the long-standing issues of the traditional networking model, then illustrate how NFV/SDN provides promising opportunities to overcome those issues. A set of evaluation metrics is proposed in order to make a comparison between the traditional networking model and NFV/SDN model. The key findings are summarized in Table 2.1. Second, the NFV architecture and its key functional components including SDN are presented, illustrating the key concepts of NFV. For example, how network functions can be migrated from dedicated hardware appliances to the virtual machines, how they are deployed and orchestrated, and how they can be chained together for achieving end-to-end network services. Third, a comparative study on the existing NFV management and orchestration (NFV MANO) frameworks is conducted. The objective is to identify whether the existing frameworks provide the capability of achieving autonomic security management, is there any security APIs available, and how they are defined and specified. Fourth, a conceptual design framework of Security Management and Orchestration (SecMANO) is proposed to improve security management in NFV environment. The chapter is concluded with some remarks.

2.1 SDN and NFV Driven New Networking Paradigms

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are two closely related technologies that dominate the evolution towards next generation networks. Both of them essentially rely on software based approaches, while operating at different levels of the network. SDN allows programmable networks by decoupling the control plane from the data plane, enabling network providers to have a better control and faster configuration. Meanwhile, the objective of NFV is to consolidate multiple network functions onto software, which run on a range of industry-standard hardware. As such, they can be easily migrated to various locations within the network on demand, greatly eliminating the needs to purchase and install new equipments. It can help to manage rapid demand growth, while reducing Capital Expenditure (CapEx) and Operational Expenditure (OpEx) in terms of hardware acquisition and capital investment. Also, time to market can be significantly shorten for new service deployment, with enhanced flexibility, agility, and scalability. Moreover, the network operators, who maintain the operations of network services, are allowed to dynamically deploy new resources, making network functions scale on demands, and providing opportunities to test services of interest with lower risks [117, 98, 191]. Thanks to these promising benefits, the global NFV and SDN markets will grow from $\$2.7$ billion to $\$15.5$ billion in 2020, at a robust Compound Annual Growth Rate (CAGR) of 42% [113]. In addition, it is foreseen that by 2020, 80% of overall

networking equipments will be transitioned to virtualization, while 10% of all service providers will invest on orchestration platforms, leading over $\$1.6$ billion in revenue [215].

2.1.1 Issues with traditional networking model

As a matter of fact, adding a new service into today's network is an extremely complex and tedious process. Due to the proprietary nature of existing hardware appliances (fixed functions), service agility and scalability are serious issues. Thus, high cost of operations and maintenance are commonly expected, making the configuration of network services increasingly difficult and cumbersome, especially when the size of network increases. The challenges about the deployment of network function in traditional networking model are discussed as follows.

- *Large capital expenditure:* Sherry et al. [210] pointed out that today's network infrastructure is expensive, complex to manage, and creates the pain points of network administrators in terms of difficulty. Even though the number of network devices is not very large, their deployment can be costly and require high up front investment in hardware. For example, the medium networks with 1k to 10k hosts may cost around 50K to 500k dollars, while a very large network with more than 100k hosts can cost over million dollars of hardware investment.

- *High management complexity and operational expenses:* Clearly, handling a large set of heterogeneous devices requires broad expertise and a large management team. As reported in [210], even a small network composing of only ten network devices may require a management team of 6-25 persons. Most administrators have to spend 1-5 hours per week to deal with device failures. It has been recognized that the top three most common issues are misconfiguration, overload, and physical/electrical failures. Also, as stated in [186], ensuring the traffic to be directed through the desired sequence of network devices require significant manual effort and operator expertise. This complexity stems from the need to carefully plan the network topology, manually set up rules to route the traffic across the desired network devices, and implement safeguards to guarantee correct operations in the presence of any failure and overload.

- *Non-optimal deployment of security functions:* Considering the aforementioned issues, it is important to determine where the network devices have to be installed to achieve expected security with minimum cost. In other words, the network operators have to choose reasonable security functions and deploy them into appropriate locations to achieve the best protection coverage. However, in practice, Shin et al. [213] argued that it is difficult to place security functions and network devices with unpredictable threats coming from different network's tenants. It is also a challenging task for network operators to configure network devices without introducing any errors. In multi-tenancy environment, these installed security functions may not lie in the best location to offer the best security service.

- *Lack of fine-grained control:* Gember et al. [88] mentioned abouts today's network devices have very limited configuration policies and narrow range in parameter manipulation. Also, the internal algorithms and operational states are completely inaccessible and unmodifiable. Apparently, it lacks fine-grained control (*e.g.*, capability to re-route traffic flows or understanding the behavior of network devices), scalability, flexibility, and reliability.

- *Lack of efficiency for usage and management:* As stated in [207], today's network infrastructure has been developed in a largely uncoordinated manner. A new form of network

devices typically emerge as an one-off solution to meet specific needs, then it is patched into infrastructure through ad-hoc and manual technique. This leads to serious inefficiency on two aspects.

- *Inefficiency in the use of hardware resources:* Network devices are typically resource intensive, each of them is usually deployed as stand alone device and independently provisioned for peak load. These resources are hard to be amortized across applications even though their workloads offer natural opportunities to do so.
- *Inefficiency in management:* Each type of today's network devices has its own customized configuration interfaces, without providing a common standard or tool to offer network administrators a unified view for managing and controlling these network devices across the networks.

Clearly, it can be observed that the aforementioned issues will be increasingly problematic for the deployment and management of network services, considering the fact that these services are continuously growing both scale and variety, while a large set of diverse service components and heterogeneous elements are incorporated at multiple layers.

2.1.2 Principles of NFV and SDN

Network Function Virtualization (NFV). As aforementioned, the emergence of NFV [67] drives massive changes in networking paradigm, thereby significantly changing the way that network services are deployed, managed, and operated. In particular, the NFV concept was first proposed by the European Telecommunication Standard Institute (ETSI) with the purpose to reduce hardware investment cost, enhance capacity of resource utilization, and accelerate service deployment of new network services to support business revenue and future growth objectives. As shown in Fig. 2.1, NFV aims to decouple network functions (*e.g.*, firewall, proxy, IDS/IPS) from dedicated hardware devices using virtualization and cloud technologies, and abstracting them into softwares known as Virtual Network Function (VNFs) running over the virtual machines (VM). There is a possibility that a single VNF can either run on a VM as a $1:1$ mapping model, or decomposing into smaller components as a $1:N$ mapping model with the goal to improve a better scalability, reusability, and faster response time (*e.g.*, [246, 58]). Also, a diverse set of VNF instances can be chained together, so-called Service Function Chaining (SFC) to create on-demand network services with regards to the particular needs of the users specification (*e.g.*, [186, 142, 146]). Thanks to these features, network services can be created, configured, scaled, and terminated quickly to accommodate with the dynamic changes in user requirements [191, 98].

Among a variety of NFV use cases documented in [72], virtualizing the functionalities of Customer Premises Equipments (CPE) (*e.g.*, routing, firewall, VPN termination, IDS/IPS, DPI) and Evolve Packet Core (EPC) (*e.g.*, Mobility Management Entity (MME), Serving/Packet Gateway (S/P GW), Home Subscriber Server (HSS)) are the most important use cases that have attracted a lot of attention from both academic and industry communities. Network operators have seen the potential benefits brought by NFV including reduction of their CapEx and OpEx costs, better flexibility of management, dynamic scaling of resources, services agility, which hence increase their revenue.

Software Defined Networking (SDN). It is considered as a new networking paradigm that promises to transform typical enterprise networks (which have static, hardware-centric, and device-specific network) into programmable, software-defined, and centralized control networks, allowing network operators to adjust and program network resources dynamically to

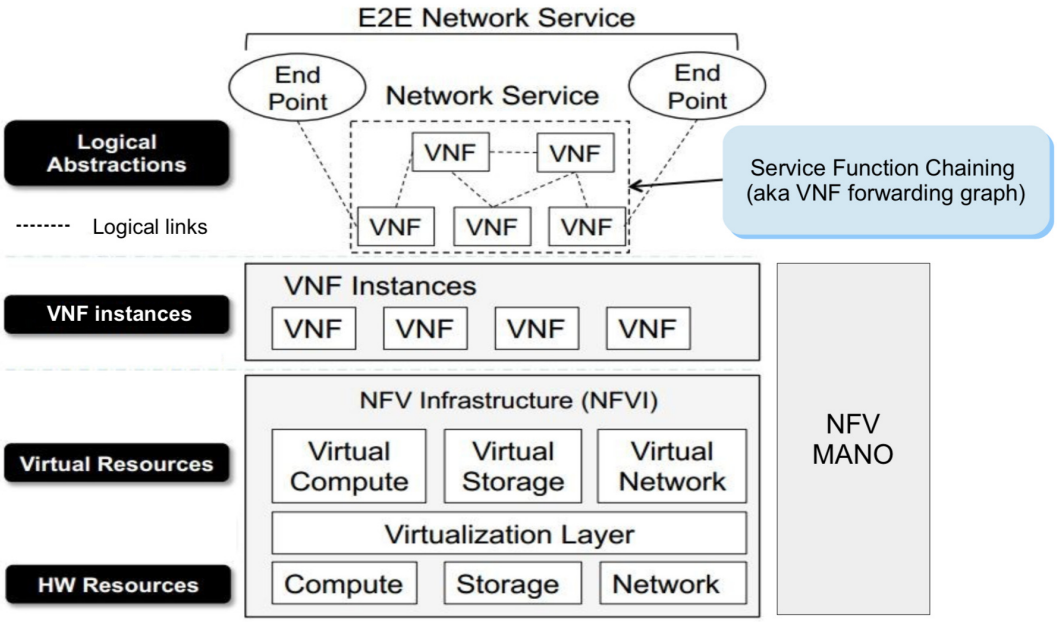


Figure 2.1: High-level NFV framework

meet the changing needs of today's business-driven networks [237]. SDN makes easier to create new abstractions in networking, simplifying network management and facilitating network evolution [126]. In particular, SDN was devised by researchers who were frustrated by the need to upgrade or change out software in network hardware devices every time they wanted to try something new. As such they introduced the idea of SDN to break down the vertical integration, by detaching the control plane from the underlying forwarding plane and using well-defines interfaces to enable programmable behavior of the network and its elements. Additionally, the concept of centralized controller is promoted with the purpose to maintain the global view of network status and operations. The controller can query all the flow entries across the network to identify individual traffic paths, request per-switch statistics of the ports as well as flow utilization. Also, it can build a full topological representation of the network, allowing re-routing decision to be made on the fly. Combining all the data available at the controller, it is possible to have a fine-grained view and control of the network utilization. To elaborate the functionality of SDN architecture, it can be simply divided into three main layers/planes as represented in Fig. 2.2:

- *Application layer:* which contains various SDN applications for various functionalities such as routing, load balancing, and security services. It communicates with the SDN controller through northbound interface, *e.g.*, RESTCONF [3].
- *Control layer:* make decisions about where traffic is sent. It consists of one or more SDN controllers. It is responsible for logically maintaining a global view of network, providing hardware abstraction to SDN applications, and performing control tasks to manage the networking devices in the infrastructure layer via southbound interfaces (*e.g.*, OpenFlow [164][1]) according to the requests from the applications. Some examples of active and open-source SDN controllers are OpenDaylight [225], Floodlight [19], ONOS [224], RYU [159].

[1] In particular, ONF [166] defines the OpenFlow as the first standard communication and the most widely-used protocol defined between the control and data planes of an SDN architecture.

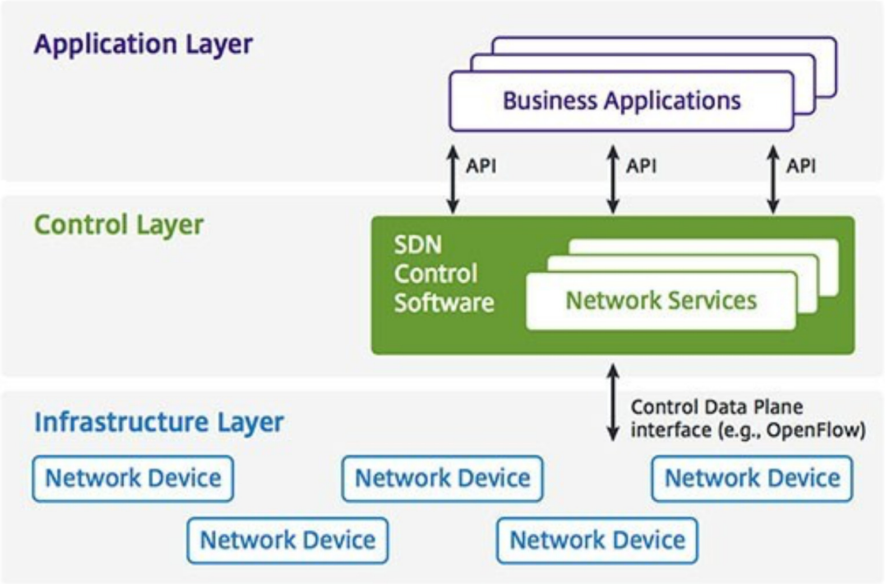


Figure 2.2: A graphical representation of SDN architecture

- *Infrastructure layer:* is composed of networking elements such as virtual/physical switches and routers that used to forward traffic flows based on rules installed by the controllers. It enables data transfer to/from the end hosts and manages conversations with the remote peers. Examples of SDN enabled OpenFlow switch implementations are Open vSwtich [134] and Pica8 [182].

2.1.3 Major advantages of NFV and SDN

It has been widely recognized that the new networking technologies like NFV and SDN bring many advantages such as reducing the cost of hardware investment, optimizing resource consumption, and improving the operational efficiency and the quality of service deployment. Therefore, the aforementioned issues in the traditional networking model can be potentially overcome. The improvements made by NFV/SDN[2] are discussed as follows.

- *Reduced CapEx and OpEx:* One of the major advantages of NFV is to reduce CapEx and OpEx, by decoupling network functions (*e.g.*, firewall, load balancing, IDS/IPS) from the proprietary hardware appliances to virtualized network services [98, 191]. As such, network functions will no longer be tied to a particular hardware platform, allowing them to be controlled centrally, and dynamically deployed on demand. Meanwhile, SDN [237, 206] which is recognized as the complimentary technology to NFV helps in overcoming the limitations associated with traditional network infrastructure playing an important role in managing and controlling the network connectivity. These advantages significantly contribute to the simplified and automated operation, eventually leading to the reduced capital investment costs and enhanced capacity utilization.

- *Shorten time to market:* Time to revenue is another key aspect provided by NFV and SDN [107, 114, 110]. It enables to deploy new services more quickly and easily (*i.e.*,

[2] NFV and SDN are two closely related technologies that operate at different layers of NFV architectural framework. SDN is considered highly complementary to NFV, which is practically used as a part of SFC creation to provide the full network control capabilities and manage traffic steering at the data plane. They usually come together. Therefore, in this thesis, we generally threat SDN as part of NFV, so that SDN will not be explicitly mentioned unless it is particular needed.

from months to minutes), while providing more scalable service creation, rapid prototyping and testing, remote and automated software update, flexible and adaptive network design tailored to particular business needs. These properties reduce the time required for instantiation, configuration, and deployment of new services, resulting in shorter service deployment time.

- *Reduced complexity of deployment and management:* Thanks to NFV management and orchestration [74, 152], network operators can remotely configure, automatically provision, and easily update policy rules of deployed network services on the fly. It offers simplified policy enforcement by eliminating the need of manually planning on device placement or pre-configure route to enforce policies. Also, with the help of centralized SDN controller [237], it provides network engineers and administrator to have a global view of network status, enabling them to directly and programmatically configure the underlying forwarding plane like switches on how to efficiently handle the network traffic. It brings efficient management of network resources by reducing overall management time and the chance for human error.

- *Dynamic and elastic scaling of services:* NFV/SDN brings significant advantages that allow service providers to scale up or down network service's capability on demands. For example, the network operators may leverage NFV/SDN to scale additional resources for particular service instance when the number of users increases, and this service instance will be removed if it is no longer used. As such, service request response time can be significantly improved, and the service configuration and updates can be handled in a faster manner. Some existing examples of NFV/SDN based network service implementations which provide dynamic scaling are discussed in [248, 58, 57].

- *Efficiency improvement for usage and management:* As aforementioned, today's network devices are developed in a largely uncoordinated manner, and it is difficult to properly operate among each other to meet the exact user requirements. To tackle this inefficiency problem of managing and deploying network services, Sekar et al. [207] exemplified two potential benefits. First, NFV allows network operators to consolidate network applications from different vendors to run on a consolidated hardware platform, and manage them in a centralized manner. Second, NFV together with SDN provide centralized management with an unified and network-wide view to quickly configure and monitor network services, and update the policy rules. For examples, the use of centralization to simplify network management has already been introduced in [248, 212].

2.1.4 The role of SDN in NFV

Being born at different times and promoted by different communities and organizations, SDN and NFV share many properties and are highly complementary to each other. These two technologies have the same objective to help accelerate service agility and drive new service innovation towards a software-driven networked ecosystem [167, 69]. On the one hand, NFV can serve SDN by virtualizing SDN elements including SDN controller and SDN data forwarding entities (which can be seen as VNFs) to run in the cloud, thus allowing the optimal provision and dynamic migration of these components with respect to their locations and business needs. On the other hand, SDN can be a part of NFV by providing programmable network connectivity between instantiated VNFs, enabling optimized traffic steering and intelligent service chaining [**?**, 158].

To have a better understanding about the SDN usage in the NFV architectural framework, Fig. 2.3 shown mapping position/location in which SDN elements can be deployed in the frame-

Table 2.1: A summary of comparative analysis between traditional network model and NFV model

Metrics	Characteristics of traditional network model	Characteristics of NFV and SDN
Cost	• Specific hardware equipments perform specific network functions • The amount of hardwares are increased when network infrastructure get large	• Minimize hardware expenditure, and software development cost • Save money on electricity consumption, hardware maintenance, and operational expenses • Space saving in datacenter
Network function deployment	• Legacy networks become difficult to be automated, provisioned, and orchestrated • Traditional networking model introduces complex configuration • Require significant manual effort and operator expertise • Longer time for service deployment	• Migrate network functions from dedicated hardware to virtual machines, and move them to software-based approach • Shorten time in service deployment • New services can be added, deployed, installed, and provisioned more quickly
Scalability	• Limited capacities scaling	• Flexible scaling up/down network services and resources as on demands • Agile service creation and rapid provisioning to improve customer satisfaction
Capability of management and orchestration	• Lack of effective automated resource allocation and configuration • Each network device has individual setting and configuration • Closed and proprietary setup, making network devices complex and difficult to manage • Lack of fine grained control, flexibility, and interoperability • Requires dedicated administrators to maintain network traffic load balancing • All requests are passed through a single piece of hardware, any failures will cause collapse of the entire services	• Provide resource optimization and load balancing • Handle virtualized resources through appropriate abstracted services, known as NFV MANO • Rapid scaling and efficient allocation of physical and virtual resources on the fly • Provide automate orchestration in action, and reduce operational cost without downgrading reliability • Simplify service chain provisioning, making it easier and cheaper to spin up applications in enterprises across service provider's networks
Built-in security mechanisms	• Specific security devices are used to prevent specific attacks • Difficult in testing, monitoring, and troubleshooting a service across multiple vendor's hardware platforms	• Allow the additional layer of security with lower cost • Rapid deployment of software-based virtual security appliances • It is easier to create, manage, and adjust security zones • Quickly automated virtualized security functions as on demand, while security policy can be updated remotely

work. In the figure, SDN elements like SDN controller and SDN application can be positioned in different locations within the NFV framework. For example, the SDN controller can be either merged with the VIM functionality (case 1), serves as a part of NFVI and is not a VNF (case 3), part of OSS/BSS (case 4), or it is virtualized as a VNF (case 2) or PNF (case 5). Similarly, the SDN application can interface with SDN controller in multiple scenarios. For example,

- *Case 1:* SDN application resides within NFVI layer and directly interfacing with SDN controller, *i.e.*, OpenStack Neutron [157].
- *Case 2:* The SDN application is virtualized as VNF and communicated directly to SDN controller. For instance, a Policy & Charging Rules Function (PCRF) may communicate to the SDN controller for policy management and traffic steering [69].
- *Case 3:* The SDN application can be an element manager that interfaces with SDN controller to collect some metrics or configure some parameters.
- *Case 4:* The SDN application might be an OSS/BSS application interfacing with SDN controller, *e.g.*, tenant specification purpose.

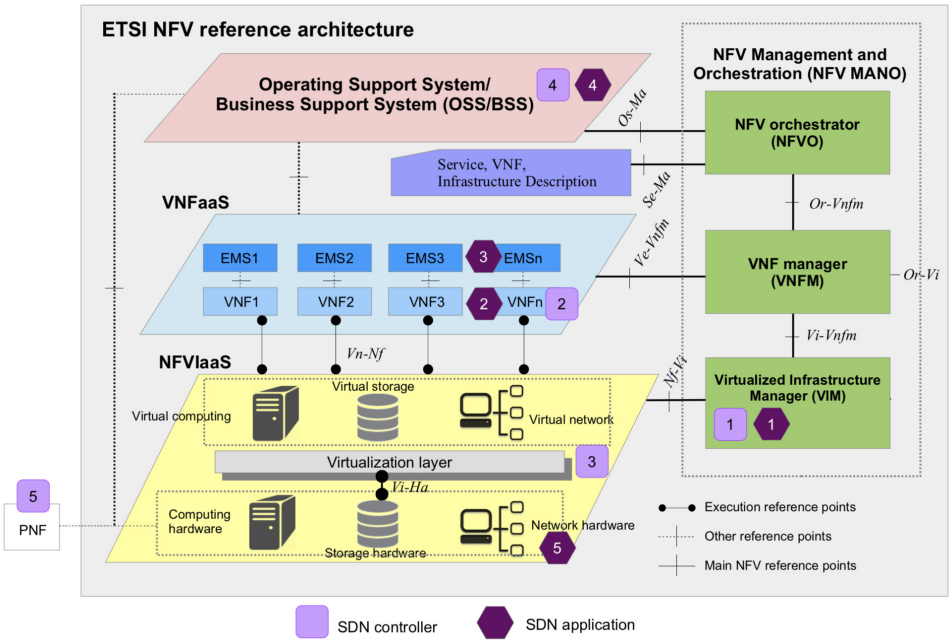


Figure 2.3: Possible locations of SDN controller and SDN applications in the NFV architectural framework

- *Case 5:* A complete SDN solution including SDN controller and SDN application are integrated in the dedicated hardware devices.

It has been revealed that the combination of SDN and NFV enables dynamic, flexible deployment and on-demand scaling of network functions which are necessary for upcoming 5G systems such as mobile core network, home environment and content delivery network. These characteristics have also encouraged and gained a promising development of network slicing and service function chaining (SFC). From a user endpoint perspective, the term of network slicing is to group physical/virtual resources into a *slice* to achieve performance requirements (transmission rate, delay, throughput, *etc.*) [158]. From network perspective, slicing a network is to divide the underlying infrastructure resources into a set of logically isolated virtual networks. This concept is considered as an important feature of a 5G network, and also being standardized by 3GPP [1]. Meanwhile, SFC [96] allows traffic flows to be routed through an ordered list of network functions (firewall, load balancers, *etc.*). Some example use cases of SFC are given in [95, 127, 233].

2.2 NFV Architectural Framework

In NFV, software-based VNFs are instantiated across a diverse range of VMs, connected and chained together in a certain way to create an end-to-end network services. According to the ETSI reference model [73], the NFV architecture is mainly composed of three main functional blocks: NFV Infrastructure (NFVI), Virtual Network Functions (VNFs), and NFV Management and Orchestration (NFV MANO), as illustrated in Fig 2.4. The additional module like Operating Support System/Business Support System (OSS/BSS), which provides management and orchestration of legacy systems, can be considered as independent module which is practically managed by NFV MANO.

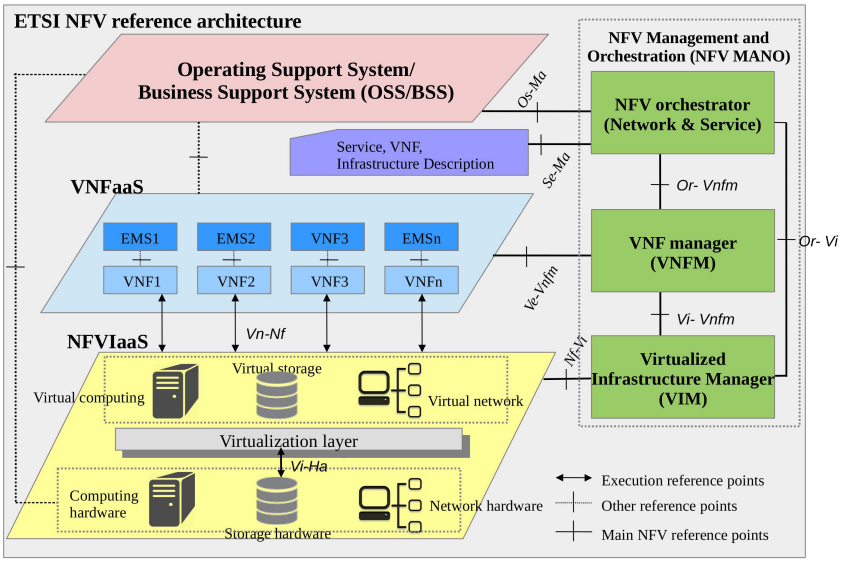


Figure 2.4: High level architecture of NFV aligned with ETSI NFV framework

2.2.1 NFV Infrastructure (NFVI)

NFVI aims at providing basic storage and computing capabilities for building an environment, in which network functions can be executed. The capabilities provided by NFVI are the processing, storage, networks, and other fundamental computing resources, which are pooled and made available to the customers. Thus, the customers are able to deploy and run arbitrary network services without too much concern about managing or controlling the underlying infrastructure. On the one hand, the concept of NFVI can greatly expand a carrier's coverage in terms of locations for providing and maintaining network services at a large scale. This brings significant advantages on reducing the cost and complexity of deploying new hardware or leasing fixed services. On the other hand, service providers can either use their own NFVI or leverage other service provider's infrastructure to deploy their own network services. This makes the implementations of network services more flexible and efficient in resource utilization.

2.2.2 Virtual Network Functions (VNFs)

This layer provides the implementations of network functions, *e.g.*, firewall, load balancing, IDS/IPS, using software based approaches. It moves network service functionality from purpose built platform to commodity hardware environment, and capable of running over the NFVI, rather than investing its own capital on purchasing specific hardware equipments and deploying network infrastructure. On the one hand, one network service can be composed by a number of VNFs that are running on several VMs, being linked and chained together to create an end-to-end network service, so-called *Service Function Chaining (SFC)*. On the another hand, a single VNF can be shared and leveraged by a number of distinct service chain to further met application-specific requirements. To date, the most prevalent transition models of VNFs, include virtual Evolved Packet Core (vEPC), virtual Customer Premises Equipment (vCPE), virtual Radio Access Network (vRAN), virtual Content Delivery Network (vCDN), virtual Set-top Box (vSTB), and virtual Residential Gateway (vRGW).

2.2.3 NFV Management and Orchestration (NFV MANO)

The nature of NFV determines that new network services can be dynamically and quickly created, deployed, and removed. However, it requires management and orchestration module to be added for managing and orchestrating all the resources needed by the VNF instances. In particular, NFV MANO has been proposed by ETSI [74], which is used for lifecycle management of physical/virtual resources, VNFs, and network services. As shown in Fig 2.4, NFV MANO can be categorized into three functional blocks as follows.

- *Virtualized Infrastructure Manager (VIM):* which manages and controls NFVI physical and virtual resources including computing, storage, and network in a single domain. In particular, the tasks related to VIM are: (1) orchestrating the allocation/request of NFVI resources; (2) performing root cause analysis of performance issues arising from NFVI; (3) collecting infrastructure fault information; (4) collecting information related to capacity planning, monitoring, and optimization; and (5) managing software images (*e.g.*, add, delete, update, copy) as requested by other NFV MANO module like NFVO.

- *VNF Manager (VNFM):* it is responsible for deployment, configuration, lifecycle management, healing, upgrading, and other element management of VNFs. The VNFM can be deployed for each VNF or multiple VNFs. The major functions of VNFM include: (1) VNF instantiation; (2) configuration; (3) modification; (4) scaling in/out; (5) termination; (6) VNF instance software upgrade; and (7) management of the integrity of the VNF instances through their lifecycle.

- *NFV Orchestration (NFVO):* which provides orchestration of NFVI resources across multiple VIMs and lifecycle management of network services. In particular, the NFVO functionality can be divided into two broad categories: resource orchestration, and service orchestration. The resource orchestration manages and coordinates the resources under the management of different VIMs. The service orchestration manages and coordinates the creation of an end-to-end service that involves VNFs from different VNFMs domains. The major functionalities of NFVO include; (1) providing topology management of network service instances (*a.k.a*, VNF forwarding graphs); (2) validating and authorizing NFVI resources requested by VNFMs; (3) managing VNF instantiation in coordination with VNFMs; and (4) providing network service instantiation and lifecycle management (*e.g.*, create, update, scale, and terminate).

2.3 Analysis on the Existing NFV MANO Frameworks

In the previous section, we have learned that NFV MANO module is considered as a core part of NFV architectural framework which plays an important role in automated arrangement, coordination, and management of complex network services. Specifically, it takes in charge of providing centralized management of resource pool, provisioning network services, maintaining the full lifecycle of VNFs and network services, with the objective to reducing the time and effort for deploying multiple network service instances on the cloud. This section is intended to conduct comparative analysis of the existing NFV MANO frameworks from security perspective, especially the open-source onces, by analyzing their architectures, data models, salient features and capabilities of managing security mechanisms, which ultimately lead to a foundation for the development of software-defined security framework.

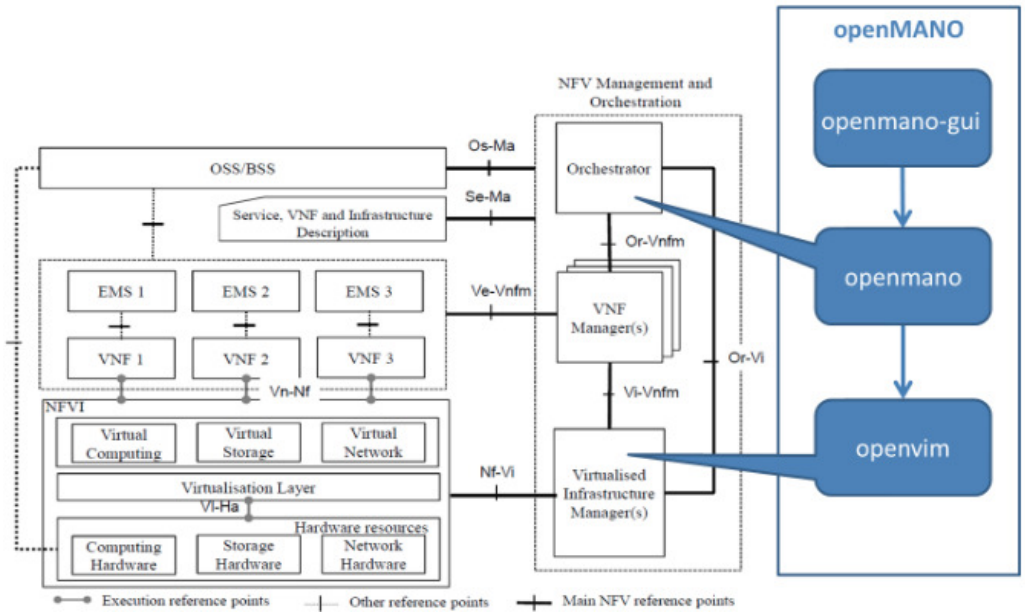


Figure 2.5: OpenMANO relation to ETSI NFV architecture [223]

2.3.1 OpenMANO

OpenMANO [223] is an open source project, which aims to provide a practical implementation of management and orchestration under ETSI standard. It has been released by Telefonica to address the main aspects of easy creations and configurations, support network service deployment in complex scenarios, and provide high performance capability aligning with Enhanced Platform Awareness (EPA) principles [68]. In addition, OpenMANO is designed to help partners and network equipment vendors to easily test and develop VNFs with vendor-neutral orchestration. As the ultimate goal of OpenMANO is to provide interoperability for service orchestration, it allows providers to adapt and expand their network service more easily.

2.3.1.1 Architecture

The OpenMANO architecture consists of three main components as presented in Fig. 2.5.

- *Openmano:* It is treated as a key component for NFV orchestrator, that supports the creation of complex virtual network scenarios. It interfaces with an Openvim through its API and offers a northbound interface based on RESTful API. The main functions of Openmano include: (1) tenant and datacenter management; (2) VNF and network scenarios catalog management; (3) VNF and Network Service (NS) deployment; and (4) VNF and NS lifecycle management.

- *Openvim:* It is considered as virtualized infrastructure manager with support high and predictable performance. It directly interfaces with the compute and storage nodes in the NFV infrastructure to deploy VMs and provide computing/network capabilities, and interacts with openflow controller to create infrastructure network topology. It offers a REST-based northbound interface (Openvim API) to communicate with Openmano for service orchestration. The main functions of Openvim include: (1) NFVI tenant management; (2) compute node, network, and port management; (3) image and flavor management; (4) VM deployment with EPA support; and (5) native and bridged layer 2 networks. In addition, the implementation of Openvim has been followed the recommendations in ETSI NFV Performance and Portability Best Practices [68].

- *Openmano-gui:* It is a web GUI, which interacts with Openmano server through its northbound API. It provides a graphical and user-friendly interface, making it easy and efficient to operate on a machine. The main characteristics of Openmano-gui include: (1) accessing to network scenario definitions and instances; (2) dragging and dropping scenario builder with access to the VNF catalog; and (3) allowing actions, *e.g.*, stop, shutdown, delete, deploy, over network service and VNF instances.

2.3.1.2 Salient Features

- *Friendly network engineer:* OpenMANO allows options to access and create network scenarios and instance lists (*e.g.*, NSs, VNFs, links) by easily dragging or dropping the NS/VNF items from catalog. The link connection among VNF instances can be created with one mouse-click. Also, network operators can freely identify options over network scenarios such as stop, shutdown, delete, and deploy NSs/VNFs. The status of all related VNF instances are available through the openmano-gui.

- *Abstraction model:* The main point of OpenMANO focuses on industry-specific orchestration challenges, by proposing an abstraction model to make network design more simple, while hiding low-level complexity of network engineer. Also, it ensures consistent deployment in which NS/VNF instances can be quickly created, configured, deployed, and terminated by referencing standard NS/VNF descriptors. In addition, network operators can utilize openmano-gui to create new network scenarios, deploy several set of VNFs and network connections, and further interact with external networks (*e.g.*, legacy nodes) to accomplish their desired tasks.

- *Enhance Performance Awareness (EPA):* One of the challenges introduced by NFV is that the virtual network functions must achieve the service capabilities and performance, as equivalent to the native network functions that have already been provided. To help address this challenge, many developers are developed their applications with specific drivers such as cryptographic or hardware accelerators [39, 29] for further improving their service performance. The concept of EPA is consistent with the objectives identified in OpenMANO, in which virtual compute nodes can be supported high performance as required. For example, OpenMANO provides properties to discover, track, report the CPU and memory usage. Also, the underlying virtual resources like vCPUs can be arbitrary allocated, intelligently deployed, and dynamically shared among VNF instances in an appropriate manner. Resulting to resource utilization and performance improvement in the available VMs.

- *REST-based APIs, and multiple VIM supporting:* OpenMANO architecture provides a REST-based northbound interface, so that Openmano module can be interacted with Openvim module for resource allocation. The use of RESTful API is a light weight communication and does not leverage much bandwidth, hence it uses a small message format for transferring. Resulting in a popular building style for cloud based APIs. Also, the concept of OpenMANO supports interoperability, as it aims to help figure out vendor lock-in. Thus, multiple VIMs such as OpenStack, VMware, OpenDaylight, OPNFV can be co-existed in Openmano layer to serve hight service performance.

2.3.2 Cloudify orchestration

Cloudify orchestration platform [47] has been proposed to help contribute open source NFV MANO adoption. It is an open source cloud orchestrator which focuses on optimizing orchestration and management of NFV based network services. The ultimate goal is to provide

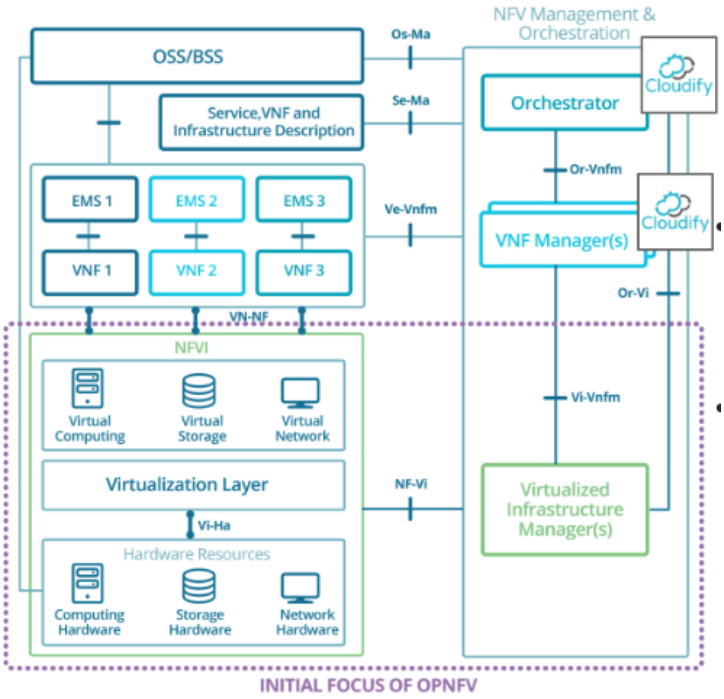


Figure 2.6: Cloudify architecture aligning to ETSI NFV MANO standard [47]

seamlessly integration with a wide range of industry standard protocols and modeling languages such as ETSI NFV MANO [74], YANG [20], NETCONF [190], RESTCONF [2], and TOSCA [227]. Thus, different workloads on the different cloud platforms can be interconnected to deliver end-to-end network services, regardless of vendor or technology stack.

2.3.2.1 Architecture

As shown in Fig. 2.6, Cloudify has been built and aligned with ETSI NFV MANO to manage and orchestrate the cloud infrastructure and resources. However, Cloudify orchestration platform is mainly focused on two modules: NFV Orchestrator (NFVO), and VNF Manager (VNFM). Meanwhile it relies on existing Virtualized Infrastructure Manager (VIM) like VMware, OpenStack, and OPNFV for handling NFVI resources.

- *NFV Orchestrator (NFVO):* as its name implies, NFVO is responsible for orchestration and management of end-to-end network services, including integration with SDN controllers and Operations Support Systems and Business Support Systems (OSS/BSS). As shown in Fig 2.7, NFVO can be composed by several modules such as, (1) REST-based northbound interface, to interact with Service portal and OSS/BSS systems; (2) Service catalog, which contains a set of blueprints for network service's execution; (3) Service orchestrator, which orchestrates and manages end-to-end network services; (4) Intelligent policy manager, provides custom policies for automated triggering, healing, and scaling capabilities; and (5) Application and network controller, provides plugins to interface with third party VNFM and other peripheral controllers such as WAN, SDN, DNS, and LDAP controllers.
- *VNF Manager (VNFM):* as similar to other orchestration platforms, the VNFM is responsible for lifecycle management of VNFs. For example, it takes action on VNF instantiation, configuration, scaling in/out, updating, and termination. Fig 2.7 illustrates that Cloudify serves as a generic VNF manager (G-VNFM) which maintain three main modules: (1) VNF catelog, which stores a set of VNF's blueprints; (2) Lifecycle management,

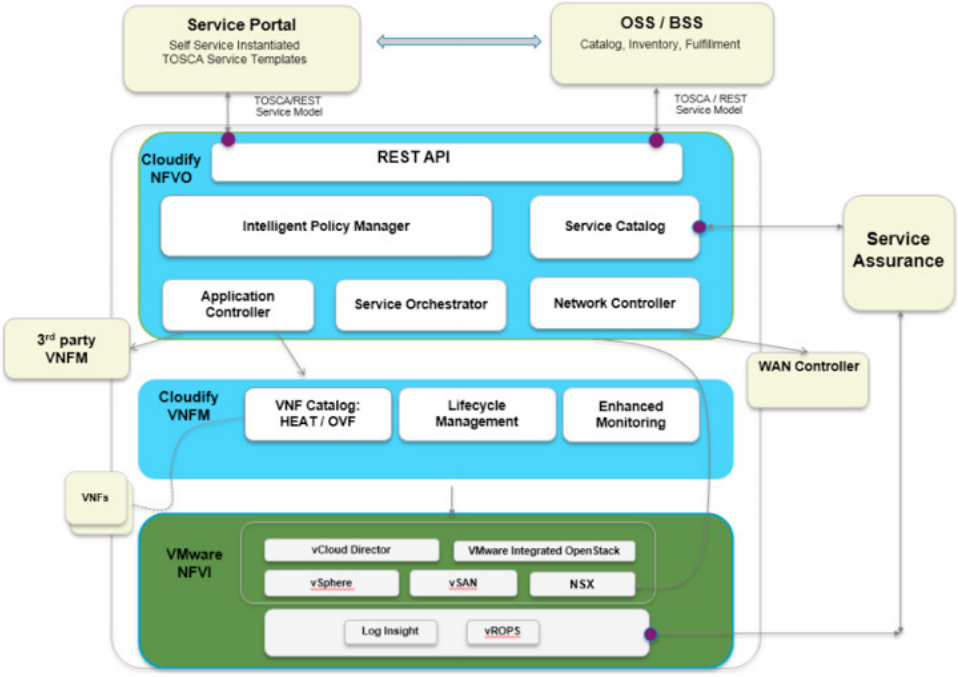


Figure 2.7: The relevant components and services in Cloudify [47]

which orchestrates and maintains the lifecycle management of VNFs; and (3) Enhanced monitoring, with an objective to provide health and performance monitoring of deployed VNFs.

In addition, Cloudify architecture is built based on TOSCA data model, which creates configuration files, called *Blueprints*, to be used for further execution of network services and VNF instances. The concept behind the blueprints aims to define the application's configuration, services, and their tier dependencies. It describes the execution plans for the lifcycle of network services. For examples, allocating the virtual resources and executing necessary middleware services needed to run network service, spinning up more instances or terminating existing ones, and performing health monitoring. Cloudify uses the blueprint as input for describing the deployment plan, making it possible to manage the infrastructure as code. As the result, it can support any cloud infrastructures, because the abstraction layer of MANO is isolated from the underlying infrastructure.

2.3.2.2 Salient Features

- *TOSCA data model:* TOSCA allows operators to arbitrary describe the deployment, operational behavior requirements, and link connection for each network service as on demands. It provides a simple way to express application topologies, VNF dependencies, and workloads in YAML syntax with a human readable fashion. More importantly, it enables portable deployment and management of applications across different cloud platforms. Resulting in service no lock-in, while providing seamlessly integration with industry networking standards and other modeling languages.

- *Native integration with OpenStack and other cloud infrastructures:* Cloudify provides an option to seamlessly integrate with OpenStack infrastructure and core services like KeyStone [171], Neutron [157], Nova [172], and Heat [104]. To do that, Cloudify re-designs its underlying architecture to match the design principle of OpenStack services. For example, rewriting the core services and leveraging the common infrastructure building blocks

such as RabbitMQ [192]. Furthermore, it provides plugin for other cloud infrastructures like VMware vSphere, Apache CloudStack, vCloud, and SoftLayer. Thus operators can deploy their services across multiple cloud environments without concerning a complex setup for particular cloud platform.

- *Multiple application supporting:* Cloudify allows operators to manage, orchestrate, and monitor a large set of applications (up to thousands of nodes). By using a message broker to manage the communication among the deployed VNF instances, while employing logging/analysis engine to monitor and keep track of that deployed VNF instances.
- *Topology-driven monitoring:* Cloudify introduces a new concept of topology driven monitoring, in which the entire application management and tracking system are centralized around the application topology. The operators can keep track at any state of applications including status of deployed network services and VNFs. Specifically, the monitoring system is integrated with the orchestration engine, making the two systems are always synchronized and up to date, so that no need to utilize or rely on the external discover services.

2.3.3 Tacker - OpenStack NFV Orchestration

Tacker [219] has been emerged as an official OpenStack project with a mission to build NFV orchestration software supporting both VNFMs and NFVO, with an ultimate goal to achieve end-to-end lifecycle management of network service and VNFs. The Tacker is designed to be compatible with the ETSI MANO architectural framework as shown in Fig 2.8.

2.3.3.0.1 Architecture: Tacker architecture consists of three major components: NFV catalog, VNFMs, and NFVO, as illustrated in Fig 2.9.

- *NFV catalog:* maintains Network Service Descriptors (NSDs), VNF Descriptors (VNFDs), and VNF Forwarding Graph Descriptors (VNFFD). All related NFV workflows and VNF descriptors are defined based on TOSCA (Topology and Orchestration Specification for Cloud Applications) templates. As a matter of fact, TOSCA is a new open cloud standard that describe software applications (including nodes and relationships) running in the cloud. In addition, the Tacker project is working closely with the OASIS TOSCA NFV subgroup [227], with the purpose to drive the evolution of its simple profile for NFV.
- *VNFM:* handles the basic lifecycle management of VNFs (*e.g.*, create, update, delete), health monitoring of deployed VNFs, auto scaling based on policy, VNF configuration using Element Management System (EMS), VNF image update management, and facilitating initial configuration of VNFs. When VNF is initiated, a TOSCA template related to VNFD is executed. In fact, a VNF can be considered as a single VM, or a complex interconnected multiple VNFs. Tacker can facilitate configuration, monitoring, healing, and scaling of VMs as described in the TOSCA template. Additionally, Tacker uses many existing OpenStack services like Heat [104] to realize these features.
- *NFVO:* provides end-to-end network service orchestration using a collection of VNFs, ensures efficient placement of VNFs based on policy, maintains VNFFD, manages resource allocation, and orchestrates VNFs across multiple VIMs spanning across different geographical sites. When network service is initiated, a TOSCA template of NSD is executed. NFVO may interact with VNFM to deploy VNF instances for a particular network service. Also, it may interact with Service Function Chaining (SFC) to further create VNF Forwarding Graphs (VNFFG).

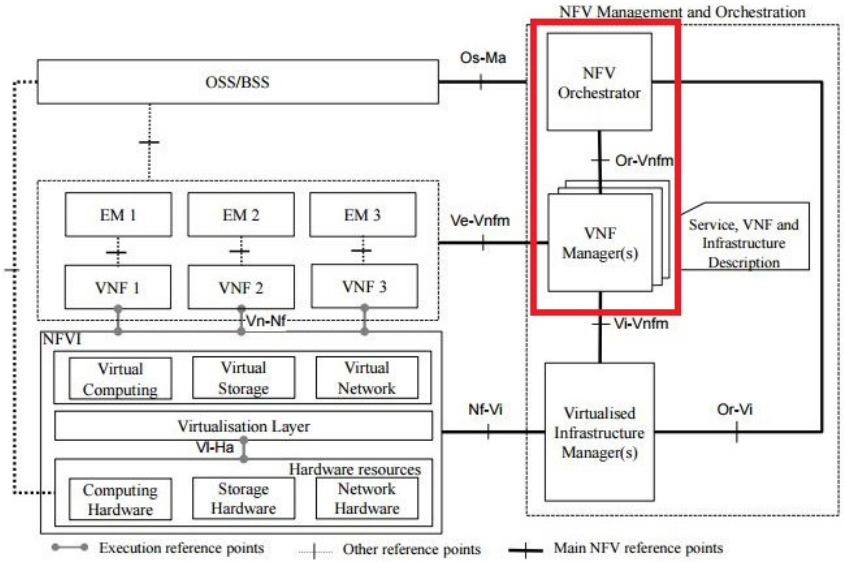


Figure 2.8: Tacker related to ETSI MANO architectural framework [219]

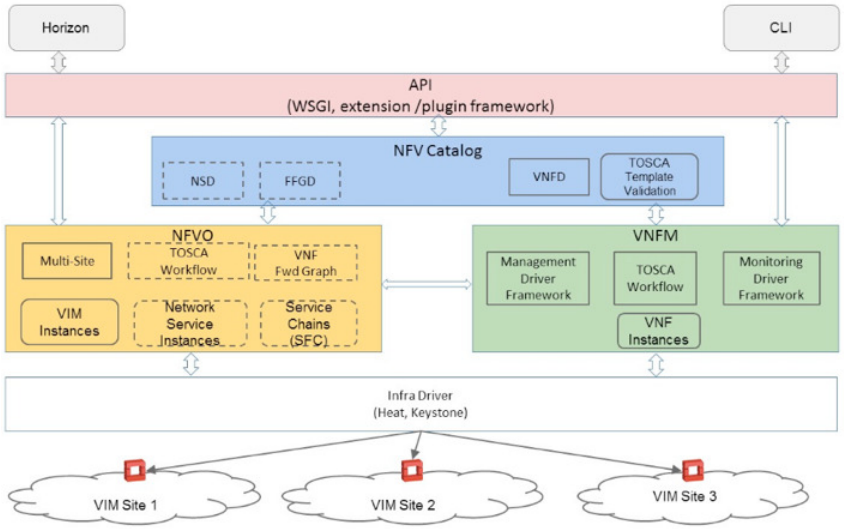


Figure 2.9: Tacker architecture with its main components - NFV catalog, VNFM, and NFVO [219]

2.3.3.1 Salient Features

- *Tacker monitoring framework:* it allows NFV operators and VNF vendors to write a pluggable driver for further monitoring status of deployed VNF instances. A monitor driver can be written using TOSCA template. For example, each VM or Virtualization Deployment Unit (VDU), which is considered as a compute node in VNF, can specify the monitor details with corresponding actions and parameters when monitoring conditions are met.
- *VNFD template parameterization:* the concept of parameterization provides ability to define a VNFD onces and used it several time for deploying multiple VNFs with different value of VM parameters providing at deployment time. Clearly, the same template can be reused several times for deploying multiple VNFs. Unlike a non-parameterized template that has static values for that parameters, this might limit the number of concurrent deployment of VNFs when using a single VNFD.
- *Enhanced Performace Awareness (EPA):* the main idea behinds the EPA is to improve overall service performance. According to Tacker architecture, it has been relied on

VNFD templates for describing a VNF in terms of its deployment and operational behavior requirements. This template allows operators to define specific requirements for particular VNF instance. It leverages features of a compute node such as Non-Uniform Memory Access (NUMA) topology, Single Root I/O Virtualization (SR-IOV), huge pages, and CPU pinning to achieve EPA's objectives (*e.g.*, high performance and low latency requirements).

- *Multi-site VIM usage:* a Tacker orchestrator can be used to control and manage multiple VIMs spanning across different OpenStack sites, without having the need to deploy a dedicated Tacker server for each OpenStack site. This allows operators to arbitrary deploy VNFs across multiple OpenStack sites using the multi-site VIM features. For example, once the operators are successfully registered on some specific OpenStack site, they can deploy VNFs globally across multiple VIMs belonging to this Tacker orchestrator.

2.3.4 OpenBaton orchestration

OpenBaton [168] project aims to provide several components for building a complete environment fully compliant with the ETSI NFV MANO specification. The main scope of OpenBaton is similar to orther NFV orchestration frameworks including OpenMANO, Tacker, and Cloudify, which particularly focuses on the basic orchestration of NFVO and VNFM, enabling VNF deployment on top of multiple cloud infrastructures. It provides a NFV Orchestrator (NFVO) to expose a dashboard for managing network services, supports a generic VNFM and a lightweight Element Management System (EMS). Also, it can integrate with a multi-site NFV infrastructure based on OpenStack [170] and Docker [61].

2.3.4.1 Architecture

Open Baton provides a reference implementation of NFVO and VNFM based on ETSI NFV MANO specification, implemented in java using the spring.io framework, as represented in Fig. 2.10. It consists of two main components, a NFVO and a generic VNFM. More specific details of Open Baton architecture can be illustrated in Fig. 2.11.

- *NFV Orchestrator (NFVO):* The main responsibilities of NFVO is to provide end-to-end network service orchestration and maintain its lifecycle. For examples, service initiation, configuration, termination, global resource and policy management, validation and authorization of requests for NFVI. The NFVO supports two different mechanisms when interacting with generic VNFM, by using a message broker as an intermediary for messaging, or exposing JSON-based RESTFul API.

- *Generic VNFM (together with the Generic EMS):* It has been designed to take care of VNF lifecycle management such as instantiating VNFs, interacting with NFVO for resource allocation, providing VNFD and virtual machine images, instructing OpenBaton EMS to execute specific configuration scripts on particular virtual machines (VNFs). It works as intermediate component between the NFVO and the VNFs (the VM that installing VNF software on top of its). To complete the lifecycle of VNFs, the Generic VNFM interoperates with the EMS (which acts as an agent located inside the VMs), with the purpose to perform executing scripts contained in the VNF's package. In addition, the Generic VNFM can be assigned to support a single VNF, or multiple VNFs simultaneously. Meanwhile, the communications between NFVO, VNFM, and EMS have been relied on AMQP (Advanced Message Queuing Protocol) based standard messaging system - RabbitMQ [192], which is standard interface for communication.

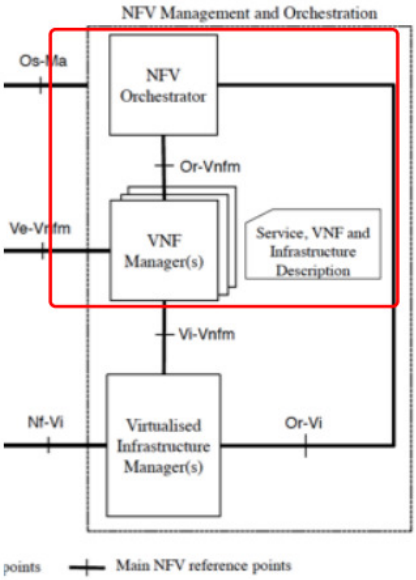


Figure 2.10: OpenBaton architecture based on ETSI NFV MANO specification [168]

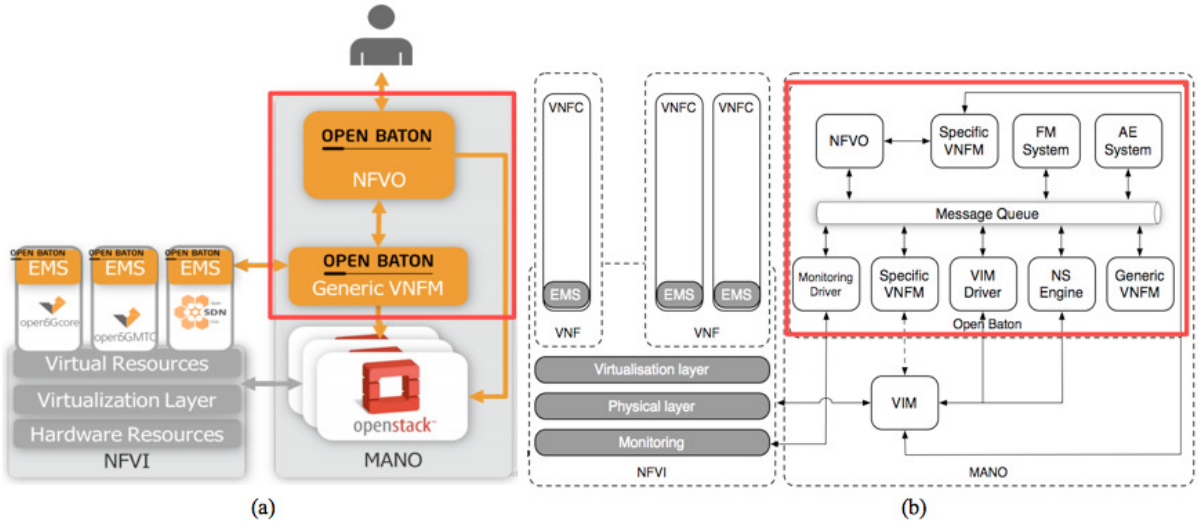


Figure 2.11: The overview of OpenBaton, (a) OpenBaton architecture, (b) the sub-components within OpenBaton architecture [168]

2.3.4.2 Salient Features

- *Openness:* To date, the works related to NFV MANO is under implementation and development, while NFV standard processes are in process and not yet mature. Interoperability among different NFV vendors is one of the key challenges that OpenBaton is trying to solve. Therefore, one of the main objectives of OpenBaton is to provide open source implementation of NFV orchestration platform, as the developers believe that this solution can help to support and improve the development of a standardized NFV environment.

- *Interoperability:* OpenBaton supports interoperability between NFVO, Generic VNFM, EMS, and other external components. As a matter of fact, OpenBaton architecture has been designed to support different mechanisms for exchanging communication between them, either over a message bus using standard messaging system (RabbitMQ), or using a RESTFul interface.

- *Easy extensibility:* OpenBaton offers easy integration and extensibility with a multi-site OpenStack environment. It uses the OpenStack APIs to initiate VM and deploy networking resources. These plugin mechanisms have been developed under OpenBaton framework to provide interface between NFVO and OpenStack VIM, and further support multiple cloud systems.

- *A set of libraries:* Open Baton offers a set of libraries, called *openbaton-libs*, that can be used to create customized VNFMs. The openbaton-libs contains several modules, sharing among different components inside the OpenBaton framework. For example, (1) NFVO uses openbaton-libs to facilitate library preparation needed for setting, installing, and instantiating different objects; (2) a generic VNFM uses openbaton-libs to deploy VMs, execute script, and perform VM's termination; and (3) Openbaton client implies openbaton-libs for further creation of NSD, VNFD, and VIM instance.

- *Dashboard:* The OpenBaton's dashboard allows the users to easily manage the lifecycle of different objects, *e.g.*, number of network service records, VNF instances, VMs, VNFD and NSD.

2.3.5 Comparative studies

Table 2.2 highlights a high level comparison between four different types of existing frameworks related to NFV MANO, especially the open-source software development.

Table 2.2: The main difference between four NFV service orchestration platforms

Characteristic analysis	OpenMANO [223]	Cloudify [47]	Tacker [219]	OpenBaton [168]
Data model	Python-based	TOSCA	TOSCA	Java based SDK
Architecture	Openmano is responsible for VNF/NS lifecycle management, it interacts with Openvim for VMs deployment, and provides computing and network capabilities	Cloudify manager acts as orchestrator which creates multiple deployments, support different plugins (*e.g.*, Docker, Chef, Puppet), execute healing and scaling, view metrics and application's topology through web UI	Tacker uses TOSCA templates to describe VNF meta-data definition. When VNF is initiated, a TOSCA template of VNFD is executed	OpenBaton provides NFVO to maintain end-to-end network service orchestration, and generic VNFM (together with generic EMS) for taking in charge of VNF lifecycle management
Centralized point of control	Yes, via Openmano	Yes, via Cloudify manager	Yes, via NFVO	Yes, via NFVO
Support OpenStack	OpenMano provides an OpenStack northbound interface (openvim API) to support OpenStack infrastructure	Cloudify works natively with OpenStack, but it is considered as an external component	Tacker is an official OpenStack project building NFV orchestration to support both NFVO and VNFMs	OpenBaton provides openstack-plugin to allows NFVO allocates resources on OpenStack
Data model based security efforts	No	No*	Policy-based VNF monitoring	No
Script terminology	Python scripts	Blueprints	TOSCA templates	Java scripts

***** Cloudify uses SSL to create secure communication between users and Cloudify manger, while relying on RabbitMQ TLS feature to create secure communication between the different components within the Cloudify.

Specifically, two important conclusions are drawn as follows,

- **TOSCA data model based implementation.** It is worth noting that only two existing frameworks (*i.e.*, Tacker and Cloudify) define their data models based on model-driven

structure using TOSCA standard[3]. One of the reasons is that other existing frameworks were already developed and implemented before OASIS released the TOSCA data model standard for NFV (version 1.0 on Mar 2016). This constraint inevitably affects their typical data structures, pressing them to be transformed from non-model driven-based approach to model driven structure using TOSCA standard, leading to error prone and cumbersome processes. An alternative solution implementing by underlying OpenStack infrastructure platform is that it developed TOSCA parser for interpreting TOSCA data model to native data structure which can be understand and recognize by the native orchestrator like Heat [104]. However, to achieve a fully model driven-based approach especially for NFV orchestration, it requires a period of time for improving maturity of NFV functional modules in terms of implementation, development, service deployment and orchestration, interfaces and standardization.

- **Security perspective.** After a careful study about the data model of these existing frameworks with respect to several core functions, *e.g.*, network topology, node specification, 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 the given frameworks generally describe network topology, node specification, and link relationships between the nodes.

  One of the key observations is that these typical TOSCA based service templates do not provide well-defined data models for security management purpose, nor clearly specify security attributes for each VM/VNF in advance. For instance, Tacker only defines the template for VNF monitoring, *e.g.*, the utilization of VNF's CPU. However, 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.

  Despite the fact that most existing frameworks are interoperated and dependent on the underlying OpenStack infrastructure orchestration service (Heat), and it is possible to define security group in Heat, there still lacks dynamic and centralized control with high level security policy specification. Therefore, it is extremely important and meaningful to develop such a dedicated module for security management and orchestration, and further explore the potential to manage a set of policy-driven security mechanisms for achieving autonomic security management (*e.g.*, self-protection, self-configuration, self-healing, self-optimization) in NFV environments.

2.4 SecMANO: An Extension of NFV MANO for Security Management

Traditionally, security management is a tedious and manual process that involves the implementation, deployment, operation, and maintenance of diverse security functions. The security administrators have to rely on dedicated software or applications to handle security events for particular operating environments. However, as NFV is based on software-driven approach, service providers can achieve much higher degree of automated network operations, independent generic hardware layer, agile application development and deployment model. Especially, thanks to NFV, operational processes such as service deployment, on demand resource allocation, on-time recovery, failure detection, and software upgrades, can be programmed and

[3] Generally, TOSCA (Topology and Orchestration Specification for Cloud Application) is considered as an OASIS (Organization for the Advancement of Structured Information Standards) open standard, which describes the structure of composite applications as topology, services, relationships, components, requirements, capabilities, and processes. The objective is to provide portability and automated management across different cloud providers.

executed with minimal time or no human intervention (zero-touch). All of these benefits provide an opportunity to reduce the complexity of configuration and maintenance. In this section, we mainly discuss security management and orchestration in NFV.

2.4.1 High-level features

The design purpose of SecMANO is four-fold,

- First, it provides the capability of integrating security functions into NFV orchestrator for improving security in network services. The dedicated hardware based security functions, *e.g.*, firewall, access control, IDS/IPS, DPI, can be migrated to NFV, so that they can be dynamically configured and automatically deployed according to the particular security requirements and user demands.
- Second, it potentially provides automated security orchestration for the entire network, making complex security solutions easier to be performed.
- Third, it allows service providers to incrementally add new functions, providing them the openness and flexibility to operate with the existing security tools via management and orchestration.
- Finally, it enables seamless automation and orchestration across multiple cloud platforms, leveraging potential security functions to improve overall security performance of network services and VNF appliances, while allowing dynamic scaling (in/out) based on the network load to serve multi-tenancy demands.

Our analysis clearly indicate that developing such a dedicated security management and orchestration module needs non-trivial efforts. The desirable features include, but not limited to, the following,

- *Advanced security service deployment:* Migrating network services like security functions from dedicated hardware devices to software-based environment allows network operators to achieve high-level agility and efficient service deployment. As reported in [64, 130], SecMANO provides different types of security as a service, so that service providers are offered with powerful forensic analysis capabilities and fast disaster recovery solutions, enabling them to launch security service (*e.g.*, intrusion detection tools) on demands. Specifically, the scope of service covers fine-grained access control to resources, global visibility on the information flows, automated security assessment and remediation, easier network isolation that isolates unstable or compromised elements from other appliances through network security zones. More interestingly, empowered by automation and central management, a set of diverse security functions can be encapsulated as on demand security services, and further be holistically integrated or orchestrated, achieving dynamic and flexible multi-layered defense mechanisms.
- *Simplifying service architecture:* As network functions have been implemented via software and migrated to virtualization environment, so that the number of hardware devices can be reduced, naturally improving physical security since there are fewer devices and fewer data centers for taking control.
- *Interoperability:* The use of NFV infrastructure, the development of virtualized interfaces, and the common protocols will allow to integrate multiple virtual appliances from various vendors with different hardwares and hypervisors, thus a large set of security functions is allowed to be virtualized and provided as agile security services by different service providers.

- *Improving capability of defending against massive attacks:* One of the most significant advantages offered by NFV is the capability to help improve DDoS mitigation, thanks to the dynamically provision based software, intelligent configuration in the system, and well-defined optimization detection. In fact, NFV enables network operators to specify bandwidth threshold and flow information for each virtual router, while the meta data associated with a particular flow can be used to analyze DDoS attack. The related example has been given in [82], by monitoring virtual routers whether there is any large amount of traffic load hitting a victim nodes that exceed the threshold value, thus DDoS attacks can be detected.
- *Automation and central management:* The architecture may facilitate consistent policy configuration and easier regulatory compliance, thanks to the automation and central management of security functions. In particular, centralized security management allows security functions to be configured according to the common policies, rather than configuring them based on a collection of network-specific security function procedures that may not be consistent and up to date. For example, patch management and incident response can be implemented in an automated and centralized fashion [130], significantly reducing operational complexities and costs. An upgraded version of network functions can be launched and tested, while the original instance remains active until the upgraded version is deployed. Also, automated incident response can be achieved, because of the inherent flexibility of NFV, enabling rapid and flexible re-configuration of virtual resources. For example, if one network function component is compromised, a cleaned version can be automatically instantiated to replace them, while the compromised version can be revoked and used for further forensic analysis.

2.4.2 The conceptual design framework of SecMANO

As NFV is still in the early stage of development and deployment, most of the existing frameworks (as discussed in the previous section) are mainly focused on the migration of network functions from dedicated hardware appliances to virtualization environment, and the effective solution in management and orchestration of the virtualized network functions. For example, the way to allocate virtualized resources, deploy and configure VNF instances, and create a service chain to deliver end-to-end network services. However, our studies reveal that security concerns are comparatively overlooked by the research communities and have not been taken into account in their development lifecycle. Although the authors of [116, 7] have already raised the security concerns in NFV orchestration and purposed security orchestrator to carry out security tasks, *e.g.*, managing security mechanisms, the detailed data models and the related use cases were not sufficiently conducted.

We therefore proposed a conceptual design framework of SecMANO, as shown in Fig. 2.12. The objective is to empower the current NFV orchestrators to manage security mechanisms, so that the most appropriate security functions can be dynamically and optimally deployed. In particular, there are two possibilities of developing and implementing SecMANO; either built-in or add-on module. However, to make it compatibility with the existing frameworks and alignment to ETSI NFV reference architecture, we propose to expand the scope of NFV MANO [74], by adding an additional module of security extension, called as *security orchestrator* to provide dynamic security management and orchestration. It works together with the NFV orchestrator to manage security functions and provide policy enforcement. In particular, security orchestrator is responsible for managing security as a service and validating security characteristics of network services and resources. Meanwhile, policy enforcement plays a significant role in controlling and managing the behavior of various VNF instances. Specifically, SecMANO have been designed with two main aspects.

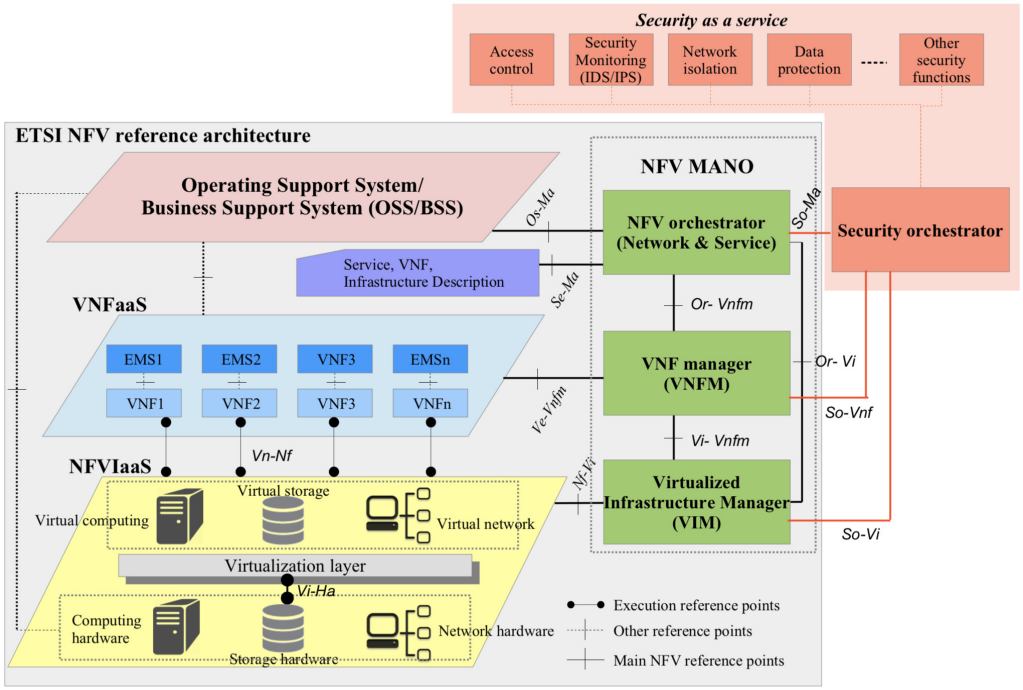


Figure 2.12: High level architecture of security orchestrator which works together with NFV orchestrator and aligned with ETSI NFV MANO

- *Security by design:* aims at formally specifying security attributes of interest at the early stage of deployment to ensure that all the deployed VM/VNF nodes in NFV environment are associated with certain security attributes. To do that, we extend TOSCA data model standard (simply known as service template) [227] and use it for specifying high-level security policy and achieving fine-grained security control.

- *Security as a service:* which aims at providing a set of security functions (*e.g.*, access control, IDS/IPS, network isolation, and data protection) on demand, ensuring that the deployed resources and services are well protected based on the specified security policy. Security orchestrator uses a set of APIs to integrate several set of security functions, and then selects the most appropriate ones for handling the user requests.

The operational workflow incooperating between NFVO and security orchestration are illustrated in Fig. 2.13. As for the processes of security management and orchestration, when NFV orchestrator receives service requests from customers, it firstly decomposes the request and retrieves the necessary recipes (*e.g.*, VNF descriptor, network service descriptor, virtual link descriptor) for execution. The NFV orchestrator may interact with various functional blocks, such as VIM to allocate virtual resources, VNFM to instantiate VNF instances, and the security orchestrator to perform security operations. Especially, this security orchestrator may use a set of security APIs to interact with a set of several security functions (*e.g.*, access control, IDS/IPS, network isolation, and data protection), and then select the most appropriate ones for handling the user requests. For example, virtual access control can be considered as the first security function to be launched by security orchestrator to examine user permission and privilege whether his/her requests are granted to access the resources/services.

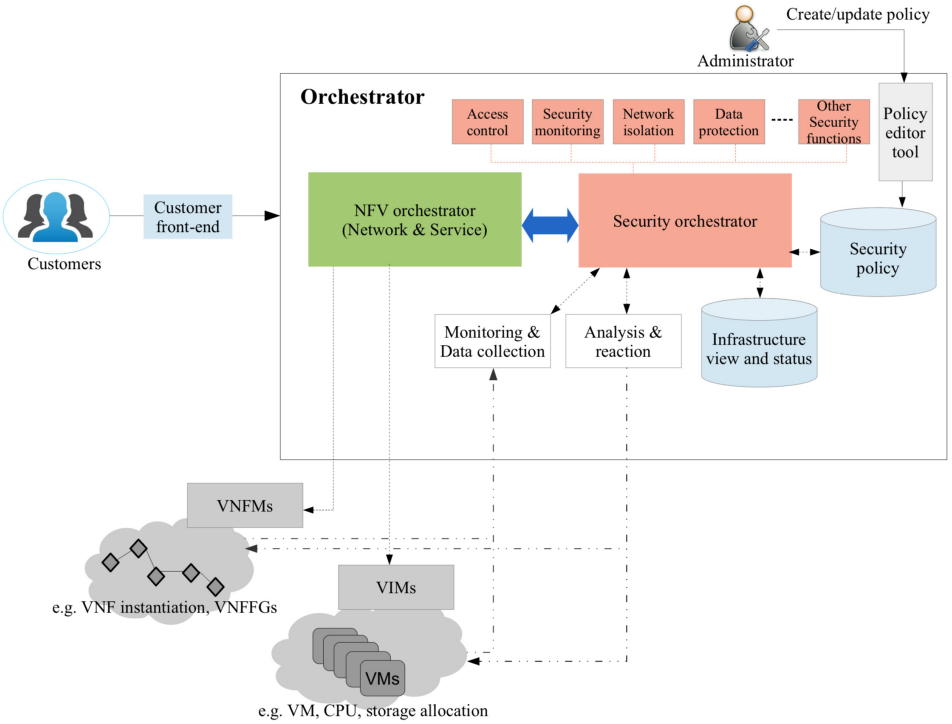


Figure 2.13: The operational workflow of NFVO and security orchestrator for service deployment use case

2.4.3 Discussions

2.4.3.1 Orchestration challenge in NFV

When network services are transitioned to NFV environment, service providers of NFV infrastructure have to deal with complexity. The challenges include a myriad of business processes and operational complexities, spanning from configuration and maintenance to management and orchestration. Recent research challenge concerns the level of intelligence that must be achieved in NFV MANO. For example, (1) it should be accomplished an acceptable level of orchestration and ensure that all the required functions are instantiated in a coherent and on-demand basis, meanwhile the solution remains manageable [191, 27]. (2) It should provide automation and self-allocation capabilities to dynamically increase/reduce the amount of resource allocation in intelligent manner [152, 18, 31]. Also, (3) the configurations should be intelligently automated at a large scale, especially those of network services that react in run time environment across heterogeneous cloud platforms [123]. As a result, the development of implementation NFV MANO framework should be smart enough to meet the above requirements, towards supporting service dynamism, flexibility, and scalability.

More importantly, a basic set of security functions should be automatically orchestrated and intelligently deployed to the appropriate NFV Infrastructure Point of Presence (NFVI-PoP) for protecting network assets based on the predefined policies. As a result, security management and orchestration need to be presented, to provide security countermeasures and offer security as a service for serving both service quality and security guarantee. In addition, it must be closely operated with NFV orchestrator to accomplish security tasks, such as providing security assessments to authenticate user requests and validating network services/resources, whether they meet certain security properties based on the predefined security policies. Furthermore, the tasks related to this security management and orchestration should be extended to provide monitoring service with the purpose to keep track of malicious events during network service and VNFs lifecycle, and towards deploying appropriate security functions in the presence of

potential threats and vulnerabilities. Nevertheless, performance impact of security management and orchestration is not yet evaluated, which would be one of the major challenges that needs further consideration and exploration.

2.4.3.2 NFV based security functions: from design to deployment

Security by design. On one hand, NFV offers promising benefits to help service providers reduce capital and operational expenses, and accelerate time-to-market be reducing the time requires to deploy new networking services for supporting dynamic changes in business requirements. On the other hand, NFV introduces a very large, comprehensive and dynamic attack surface (more details about the NFV threat surface are presented in Chapter 3). The security challenges with NFV are that: (1) there are new malwares (up to 30,000) have been occurred each day; (2) error prone processes from manual deployments and distributed policy configuration are significantly multiplied; and (3) dynamic remediation and security service assessment are not sufficiently addressed [144]. Also, Hamed et al. [97] concluded that manual configuration of network security technologies, such as firewall and IPsec on the extended sets of devices are prone to have configuration errors, intra and inter policy conflicts. This configuration complexity is one of the main reasons which generates serious vulnerabilities and threats in the network. Similarly, the authors of [40] pointed out that the challenging in configuration of distributed policies over NFV environment can be a significant administrative burden, and error prone. For instance, any update in policies can potentially lead to widespread impact. Overall, current research activities are not sufficiently addressed and taken into account of these security challenges.

Therefore, the implementation and deployment of virtual network services should be initially started from *security by design*, complemented with *security as a service*. It is generally a more realistic solution to ensure confidentiality, controllability of all the access to the underlying infrastructure assets, and ultimately improving the security as a whole. In [144], the authors discussed about security by design, mentioning that that the design process should start with enforcing zero-trust model by blocking all the network traffics until security policies are applied. For example, authenticating tenant requests, verifying certificate from VM deployment based on the requested services, and applying policy according to the tenant requirements. Also, the authors of [156] discussed four phases involving in security by design: (1) asset identification – by identifying users, service providers, and hardware assets running in NFV environment; (2) adversaries identification – identifies all the possibilities of attacks; (3) layers of defense – which provides security solutions that can be used to mitigate and defend against security threats and vulnerabilities; (4) review phase – all processes and procedures must be periodically and frequently rechecked, whether they have sufficient security protection when there are changed in user requirements or when security services are out of date.

Security as a service. As a matter of fact, many users are seriously concerned about security, privacy, and trust issues introduced by the novel networking technologies like NFV. The most common reasons are as follows. First, they are not able to monitor all the data transaction processing in the cloud environment, especially when the underlying infrastructure owned by the third party service providers. Second, they have no idea where data is, who has access to their data, how data is performed, or even how the data is transferred over the network links. Third, they are not fully trust the cloud service providers. Hence, the sensitive data of users can be presented in unencrypted form owned and operated by the third party service providers. The risks of unauthorized disclosure of their data performing by untrusted service providers could be high. The challenge is that users have to ensure the service providers do not break down their privacy and confidential data. Although ETSI has published the document related with

NFV security [76, 79], which aims at providing general guideline on how to establish a secured baseline for NFV operations (*e.g.*, security within VNFs, between VNFs, external to VNFs, and secured interfaces), it does not cover prescriptive requirements or specific implementation details. So far, several solutions have been proposed, such as the ones reported in [220, 240, 62], which aim at protecting data confidentiality and user privacy when their data is operated in the cloud. However, the management complexity and communication overhead cannot be avoided. Practically, network operators have to deal with complex tasks of operations, *e.g.*, orchestrating, managing, configuring, and maintaining the network services. Especially, when the network size gets large, containing a diverse set of network services interoperated and implemented by different NFV vendors, and spanning across multiple platforms at different geographical locations. Therefore, it is important to obtain the optimal balance between network performance goals and essential security requirements. This is in fact a long-standing challenge for any other ICT scenarios.

2.5 Concluding Remarks

This Chapter is devoted to developing a conceptual framework SecMANO for security management and orchestration in NFV environment. First of all, a broad overview about NFV and SDN technologies was presented, with a focus on addressing how SDN and NFV can overcome the issues in the traditional networking model, showing that the combined advantages of these two technologies are the driving force behind the transformation of networking architecture and various industrial products. By introducing a software-based approach and moving towards a virtualization environment, NFV/SDN makes network services become easy to deploy, allowing them to be more powerful and flexible, while enabling to create such a complex network topology and feature-rich network functions to meet specific business requirements.

Then the NFV architectural framework was introduced. The usage of SDN and its role in NFV was specifically discussed. Although NFV and SDN are two technologies that can work independently because they perform two separate tasks at different layers of network infrastructure, they complement each other which offer a new way to design, deploy, and management the network resources and services. In particular, NFV focuses on optimizing the actual network service deployment, and accelerating service innovation and provisioning, while SDN primarily focuses on the control plane to provide the full network capabilities and improve the agility of controlling the generic forwarding devices.

To develop SecMANO, we conducted a comprehensive study on the existing frameworks of NFV MANO. We deeply studied how the available frameworks (especially the open-source ones) have been developed, what features they provided, and whether they take into account security issues to improve their frameworks. Unfortunately, our studies indicate that most of them have the same objective, i.e., providing a practical implementation management and orchestration to fully comply with ETSI standard. None of them is tailored for security service orchestration, and there is no such a dedicate security orchestrator available, even though some of them take into account security functions and provide certain interfaces for managing pre-deployed security functions. Therefore, a conceptual design framework of security management and orchestration (SecMANO) was proposed, which manages security functions as a whole, allowing dynamic and adaptive configurations based on high-level security policies. As show in Fig. 2.12, it is expected to operate in parallel with a NFV orchestrator, managing the basic security functions in a holistic way.

To implement the proposed conceptual framework SecMANO and validate its feasibility and performance in real NFV scenarios, we will present a prototype development in Chapter 4. However, before doing that, we will report a comprehensive threat analysis in the NFV envi-

ronment in Chapter 3. Because we believe this will help to better identify what kind of security threats and vulnerabilities in NFV deserve to be carefully addressed, and what/how the security countermeasures can be managed by SecMANO.

3 Security in NFV

As what we have discussed in the Chapter 2, NFV and SDN provide a large pool of technological benefits, significantly reshaping legacy ICT infrastructure and pushing the evolution towards next generation network. However, one fact can never be neglected is that security remains to be one of the vital concerns and potential hurdle, as attack surface becomes unclear and defense line turns to be blurred in the virtualization environment. This Chapter is therefore devoted to investigating and exploring the potential security issues in NFV. First, we aim to analyze security threats of well-defined NFV use cases that are documented by ETSI [72], with an objective to establishing a comprehensive layer-specific threat taxonomy. Second, we conduct in-depth comparative studies on several security mechanisms that are applied in traditional scenarios and in NFV environments. The purpose is to analyze their implicit relationships with NFV performance objectives in terms of feasibility, agility, effectiveness, and so on. Third, based on the established threat taxonomy and the analyzed security mechanisms, we provide a set of recommendations on securing NFV based services, along with the analysis on the state-of-the-art security countermeasures. A resulting holistic security framework is intended to lay a foundation for NFV service providers to deploy adaptive, scalable, and cost-effective security hardening based on their particular needs.

3.1 Background

Despite NFV brings many promising advantages (as discussed in the previous section), security concerns remain to be a significant barrier to the wide adoption of NFV [178, 64, 76, 77]: novel security threats and vulnerabilities will be inevitably introduced. The resulting attack surface for NFV based network services could be even larger than its traditional counterpart, spanning from hardware vulnerabilities to the vulnerability of VNFs, orchestration, or even policy violation due to non-trivial service complexities and administrative errors. Specifically, when the large scale deployment of NFV goes across a wide range of cloud datacenters and security domains, the frequent migration can bring a large set of challenges to threat landscape identification and security policies enforcement. More importantly, as the development and deployment of network services in NFV are still in the early stage, while the technology and products related to NFV are not yet mature. As a result, it remains unclear how NFV would fundamentally impact the landscape of cyber defense, how it can help to improve security management, and what kind of specific security risks it may introduce. We envision that the attack surface of NFV could be significantly enlarged and multiplied due to the frequent migration of VMs and network service instances, dynamic formation of virtual network zones, autonomic provisioning of network services, on-demand orchestration, and so on. These factors make

NFV extremely difficult to establish in-depth defense security, requiring the critical security issues to be addressed in a holistic way.

- From a vertical (north-south) perspective, VNFs based NFV scenarios are diversely incorporated with several types of network components, ranging from infrastructure layer (*e.g.*, high volume servers, switches, and storages) to application layer (*e.g.*, VNF appliances, orchestrated modules, and service compositions). Thus, any misconfiguration of a VM instance or hypervisor could eventually allow attackers to penetrate into the whole virtual network zones.
- From a horizontal (west-east) perspective, as each layer is composed of heterogeneous elements and involved with multiple players, the trust relationship between the service components, either hardware or software, is hard to be established. It is therefore challenging to securely and seamlessly incorporate those service components from heterogeneous platforms and different vendors. More seriously, the complicated service dependencies make security policies difficult to be efficiently enforced at the most appropriate points in the fine-grained manner.

Nowadays, many academia researchers and industry practitioners have started to explore potential benefits of NFV to obtain a clear vision, *e.g.*, why they need to deploy NFV, and how it can help to reshape network infrastructure. Hence, some survey papers can be found in the literature. These existing works mainly cover the following aspects: the evolution history of NFV [133], relationship between NFV, SDN and cloud computing promising [191], implementation of each layer in NFV [230], current efforts about NFV management and orchestration (NFV MANO) [152] and research challenges related to NFV [191], *etc.* However, as discussed earlier, novel security threats and vulnerabilities introduced by NFV become one of the major concerns. To the best of our studies, an in-depth investigation on NFV threat surface has not been sufficiently conducted, the effective security solutions and countermeasures have not been systematically studied. Therefore, closing this gap is the main motivation of this chapter. We believe that this outcome can help service providers to gain a holistic understanding on the NFV's attack surface, the state-of-the-art security countermeasures, as well as a set of recommendations on securing different NFV layers from low layer (*e.g.*, NFV infrastructure) to the upper layers (*e.g.*, virtual network appliances, and business applications).

3.2 Contributions

As any other networking paradigms and services, an ideal solution is that NFV service providers and network operators have to ensure that all the processes related to network service initiation, configuration, implementation, and deployment are secure. However, this is a mission impossible in practice. Due to the intrinsic characteristics of NFV making such a mission even more challenging. A more traditional yet realistic methodology is to conduct threat analysis for specific NFV scenarios, and then identify their specific threats and vulnerabilities. The ultimate goal is to establish a comprehensive NFV layer-specific threat taxonomy. In addition, the corresponding security hardening solutions can be studied, compared and deployed for achieving cost-effective, cross-layer, and in-depth defensive line. Throughout this chapter, we intend to deliver three following contributions.

- First, we intend to conduct a comprehensive study on several NFV based use cases that are documented by ETSI [72], with the objective to analyze their specific threats and vulnerabilities. The ultimate goal is to establish an NFV layer-specific threat taxonomy. We believe that this threat taxonomy allowing service providers and network operators to

gain a holistic understanding on the NFV's attack surface, while helping them to deploy cost-effective security hardening measures according to their particular needs.

- Second, we conduct a set of comparative studies on several typical security mechanisms by comparing their implementations and deployments in traditional scenarios and in NFV environments. The considered security mechanisms are: Identity and Access Management (IAM), Intrusion Detection and Intrusion Prevention (IDS/IPS), network isolation, and data protection. Thus, the result from comparative analysis can help us to identify how does NFV impact the traditional security mechanisms in terms of feasibility, agility, and effectiveness, and how these security functions explore any potential challenges when they are migrated to NFV environment.

- Third, from the established threat taxonomy and comparative studies of the security mechanisms, we give a set of recommendations on securing different NFV layers and analyze the state-of-the-art security countermeasures. This outcome ultimately provides a clear understanding to what extent the available NFV security best practices can fulfill the security recommendations. An NFV-based holistic security framework is finally proposed.

3.3 Use Case Driven Threat Analysis

Nowadays, novel security threats and vulnerabilities introduced by NFV become one of the major concerns. To the best of our studies, an in-depth investigation on NFV threat surface has not been sufficiently conducted, meanwhile the effective security solutions and countermeasures have not been systematically studied. The service providers are expected to take extra precautions to ensure that all the processes related to their deployments of network resources and services meet sufficient security guarantee when they are migrated to virtualization environment, which is however believed to be mission impossible in practice. Alternatively, we start with an in-depth threat analysis of specific NFV scenarios, by referring the well-defined use cased documented by ETSI [72] and reorganizing them into five specific NFV based use cases, with an emphasis on examining their architectures and analyzing their security threats. The ultimate goal here is to establish a layer-specific threat taxonomy. To reduce the volume of this survey, it should be noted that we intensively discuss more detailed threat analysis for the first three use cases. For the other two use cases are quite similar to the third use case, but differ in the context they are applied (*e.g.*, mobile network or fixed access network scenarios). Each use case is based on the same analysis methodology, making the use case driven threat analysis is relatively simplified.

3.3.1 Use case 1: NFV Infrastructure as a Service (NFVIaaS)

This use case is about providing basic storage, network, processing capabilities, and other fundamental computing resources, which are pooled together and made as transparent services, to the customers, who then deploy and run arbitrary network services without managing or controlling the underlying infrastructure. In particular, the service providers can either use their own infrastructure or leverage other service provider's infrastructure to deploy their own network services. The architecture of NFVI is shown in Fig 3.1, in which one service provider (*e.g.*, SP2) can run VNF instances on the NFVI of another service provider (*e.g.*, SP1) to improve service resiliency and customer experience. In addition, SP2 is able to integrate its VNF instances running across its own infrastructure and NFVI of SP1 for creating an end-to-end network service. As the two service providers are independent from each other, the failure of one NFVI will not affect the other.

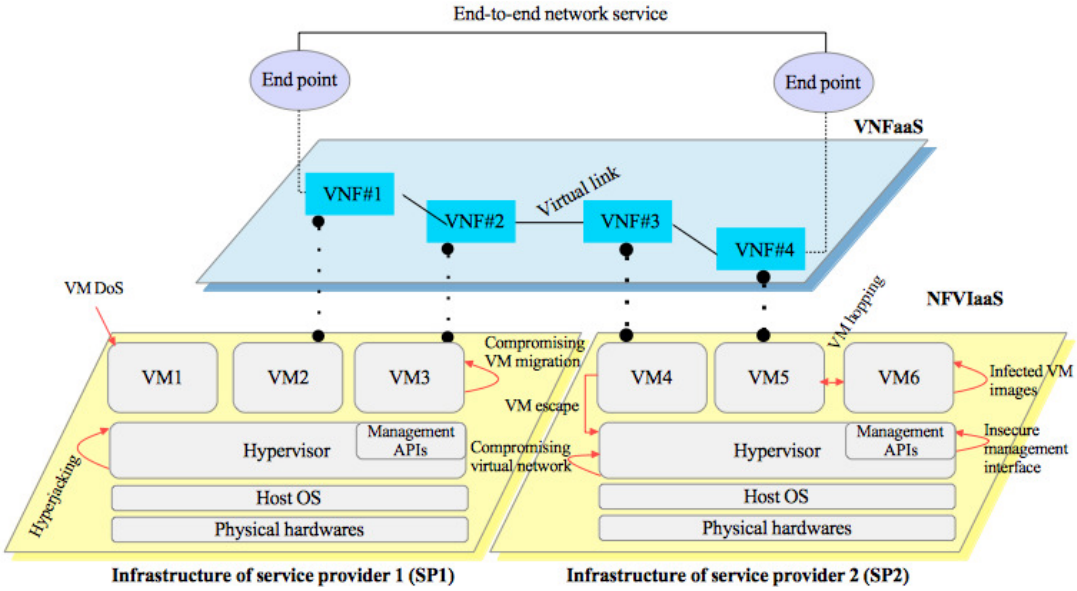


Figure 3.1: The overview of NFV architectural framework and attack models

From security perspective, the lack of security protection at NFVI layer will affect other layers and other related components including VNF instances, simply because these components run upon NFVI layer. Pek et al. [181] have identified a number of security issues in hardware virtualization, examined the potential adversaries (network adversary, local adversary) of compromising the guest VMs, host OS, hypervisor, management interface, and virtual networks. In addition to the threats of VMs or hypervisors which apparently occur in this use case, we also examine the threats that are specific to this use case.

- *Security issues in guest VMs:* As a matter of fact, the design of VMs are not conceptually different from traditional implementation of physical machines. They rely on the operating systems and provide users with virtual resources such as virtual memory and storages for further executing their desired applications. Consequently, the ability to exploit vulnerabilities in a physical environment presents a significant threat to virtualization environment as well. In the following, we exemplify threats and vulnerabilities targeting to the VMs.
  - *Infected VM images:* In particular, the VM image is a pre-packaged software template containing the configuration files which are used to launch the VM instances on demand. The network operators can either create their own VM images from scratch, or downloading VM images stored in the public provider's repository. For example, Amazon offers a public image repository where legitimate users can download or upload a VM image. However, Hashizume et al. [102] pointed out that this opportunity gives the malicious users to upload the VM images which may contain malicious code such as a Trojan horse into public repositories. In the worse case scenario, if there is any users uses this image, their VMs will be infected with the hidden malware. Also, unintentionally data leakage (*e.g.*, password, cryptographic keys) can be exposed when the malicious VM image is being created by the victim users [4].
  - *Compromising VM migration:* VM migration allows network operators to relocate or transfer any VMs from one physical machine to another. The benefit of migration significantly improve the performance of workload balancing and system management. In other words, this useful feature might be susceptible to many kind of attacks such as man-in-the-middle attack risen by sniffing the traffic, and DDoS flooding

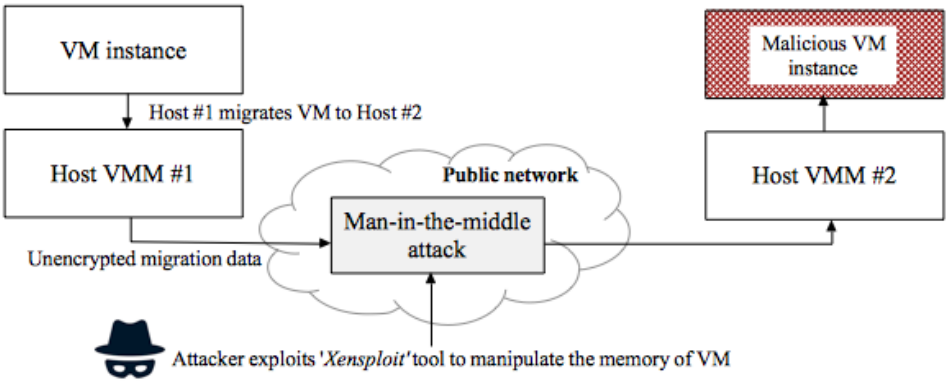


Figure 3.2: Man-in-the-middle attack scenario against a live VM migration

attack [85, 55, 181]. Due to the fact that the migration data is naturally presented in plaintext as unencrypted mode. This makes sensitive data like guest VMs can be sniffed, tampered, or manipulated easily when it has been transfered through unsecure communication channel (*e.g.*, LAN or VLAN). In [162], the authors developed a tool *Xensploit* to exemplify man-in-the-middle attack against a live VM migration. As presented in Fig. 3.2, when the Host #1 performs VM migration to Host #2, the memory pages about guest VM have been transmitted over the network and passed through the malicious node which run *Xenploit* tool. This tool allows attacker to perform illegal manipulation on the memory pages of the victim VM.

– *VM hopping:* an adversary has already gained access to a guest VM either by compromising it or hiring one in a cloud infrastructure. From a guest environment, he then compromise other guests through privileged access to the host. In [181, 217], the authors pointed out that one of the main issue leading to VM hopping is that the perfect isolations between guest VMs are violated. This could be due to an unresolved issue in the Memory Management Module (MMU) of the hypervisor that allows adversary to gain access and perform illegal manipulation on the memory pages of other guest VM in accordance with his access rights (read, write, execute).

– *VM escape:* an adversary can run malicious code on a VM, so as to gain access to the host OS and further get into other VMs running on the same host. Fig. 3.3, illustrated the attack model about VM escape. First, the adversary may compromise an application running inside a VM to gain access to its operating system. Second, using tools, *e.g.*, Cloudburst VM escape (CVE-2009-1244), he gains access to the hypervisor management interface, and then break down into the hypervisor to gain the root privilege. For example, Lal et al. [128] exemplified a practical use case of VM escape by sending crafted network packets to exploit heap buffer overflow with compromised virtualization process, resulting in the execution of arbitrary code on the hypervisor. Another practical example has been reported in [187], by exploiting a hardware simulation bug within VMware to escape from the guest OS to the host.

– *VM DoS:* As virtualization lets multiple VMs share physical resources (*e.g.*, CPU, memory disk, and network bandwidth) among each other on the same host machine to improve the utilization of physical resources. In this attack, an attacker may launch a DoS attack in one VM and try to exhaust all available resources [109, 102], or by exploiting hypervisor's misconfiguration [181]. This result leads to the situation that the hypervisor can not support more VMs, making the VMs running on the same host machine are starved for the resources.

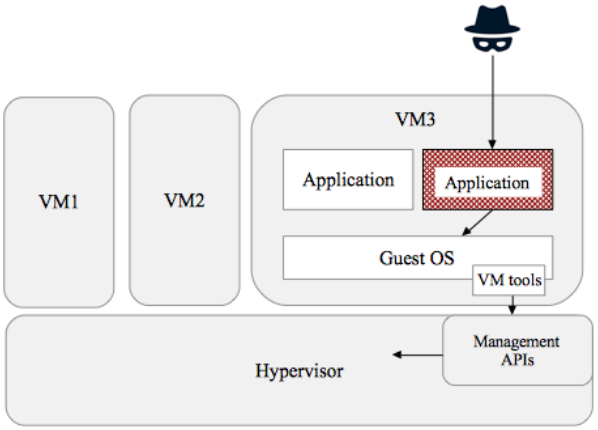


Figure 3.3: VM escape scenario

- *Security issues in hypervisor:* the Virtual Machine Monitor (VMM) or hypervisor plays a vital role for VM isolation. If the hypervisor is compromised, any attached VMs will be potentially compromised as well. In the following, we discuss two possible ways to compromise the hypervisor.
  - *Hyperjacking:* which subverts the existing hypervisor or injects a rogue hypervisor into the system, so that all the components connected to the compromised hypervisor can be manipulated [109]. As illustrated in Fig. 3.4, there are three possible methods to perform a hyperjacking attack: (1) injecting a rogue hypervisor beneath the original hypervisor, (2) running a rogue hypervisor on top of an existing hypervisor, and (3) directly obtaining control over the original hypervisor. Regular security measures are ineffective, because the host OS may not even be aware about this compromise. In fact, the hyperjacking specially operated in stealth mode and run beneath the machine, making more difficult to detect it. Although hyperjacking is rare occurrence due to the difficulty of directly accessing hypervisors, it is considered a real world threat that administrators should take into account [145].
  - *Breach of isolation:* One of the main goals provided by the hypervisor is to ensure that the hosted VMs are isolated, which means that one guest VM is not able to reach more resources than it has been granted. However, bad configuration or design flaws within the hypervisor can allow an attacker to break out of isolation, resulting in DoS attack over guest VMs, VM escape, or system halt [128, 217].
- *Insecure management interfaces:* In NFV, the management interfaces can be classified into several types, each of which may perform different functions (*e.g.*, interfaces between VIMs, or between VIMs and VNFMs). Although ETSI has published a document about NFV MANO to describe an architecture framework for management and orchestration of NFV and its interfaces [74], the specific details about interface design and implementation are not provided. Lal et al. [128] stated that an attackers may exploit the insecure interface to dump the records of user's data such as management password. Also, they may embed malicious code into the interface in order to gain the illegal access to the victim's system [75]. For example, if the management interface provides Web surfaces (*e.g.*, vSphere/VI client - central management interface for VMware ESXi [181]), a remote attacker can possibly inject arbitrary script or HTML code into it by exploring XSS vulnerabilities.
- *Compromising virtual network components:* First of all, the virtualized networking components such as routers can suffer from any software vulnerabilities, leading the whole

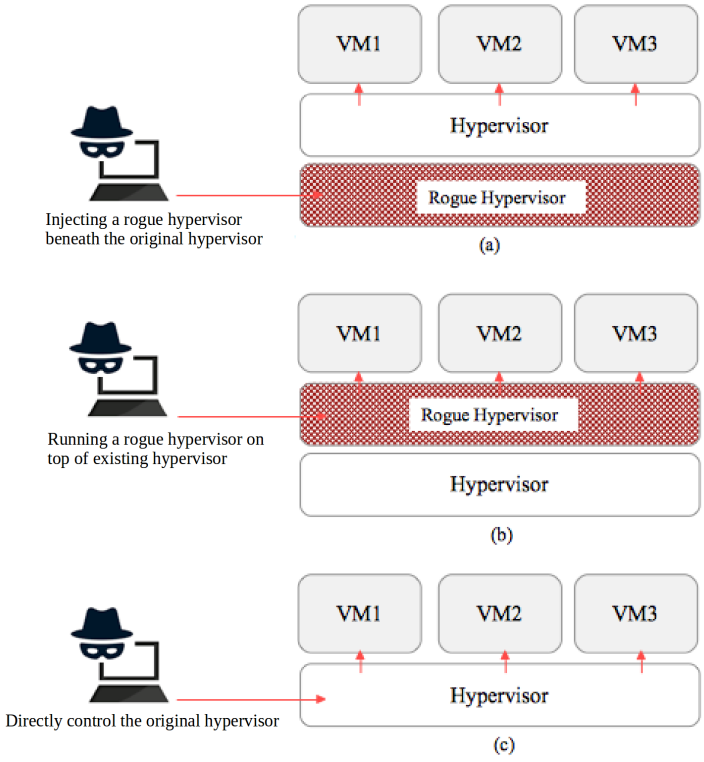


Figure 3.4: Hyperjacking attacking models

network to be compromised, *e.g.*, DoS, traffic redirection and dropping. Second, in the virtualized networks, the traditional attacks such as *e.g.*, IP spoofing, ARP spoofing, DoS attacks can still occur [55, 75], due to the misconfiguration of hypervisor, inappropriate network isolation, and the virtualized networking devices like L2 switches. For example, DoS traffic can be redirected to both virtual networks and VNF's public interfaces to exhaust network resources and impact service availability [128].

- *Security pitfalls of OpenStack:* In NFV, a majority of network service deployment and implementation rely on OpenStack platform [170]. It provides a set of software tools which are grouped together for building and managing cloud services. However, OpenStack is a relative new comer in the IaaS space (the first released in late 2010), it definitely remains many software vulnerabilities and security breaches. Hala et al. [5] investigated security issues in Openstack. For example, OpenStack utilizes the concept of projects and tenants to group users into logical units. Using administrative privilege, the user can grant access to all projects. In the worse case scenario, if the administrative password has been intercepted, then the malicious users can perform any illegal operations, *e.g.*, create new users, remove existing projects. Another security concern is that the authentication and authorization are achieved through Keystone service [171], however the current protocol still pose vulnerabilities [51]. For example, Keystone uses username and password to authenticate client. This information is sent to Keystone server in clear text using http protocol. There is a high chance of having information disclosure, an attacker can intercept username and password during the authentication process. After successful authentication, the Keystone backend generates a token and send it back to the client. This token is applied when the client need to use other OpenStack services. However, the way of sending token back to the client is not completely secure. If attacker obtains the token, he will get the client's whole privileges which would cause unimaginable disaster to the system.

- *Inadequate enforcement of security policies:* In the context of NFV, security policies specify the important components of the contractual relationship between service providers and customers, which are applied to the infrastructure as well. In particular, security policies define roles and responsibilities of all the parties (*e.g.*, administrators, end users), level of services (*e.g.*, high, medium, low), and security performance requirements (*e.g.*, clearly define the access rights of the service providers, who can access to the data, and how to mitigate data breach when it occurs). Clearly, cloud services and deployment models differ in security policies, so any misinterpretation, inappropriate enforcement, or loosely configuration of security policies will lead to security threats [55]. DoS attack is an example from in inappropriate security policy enforcement [177]. If service providers do not strictly concern about the maximum bandwidth-ceiling, attackers can take this opportunity to launch DoS attack for overwhelming the target system resources and network connections.
- *Shared physical and virtual resources:* NFV enables ubiquitous, convenient and on demand access to a shared pool of computing resources across multi-tenancy through virtualization. However, sharing the physical resources such as hard disks, RAM, CPUs, GPUs, and other elements, are not naturally well-designed to support multi-tenancy requirements. For example, sharing resources between VMs can allow malicious VM to access other victim VMs through shared memory, network connections, and other shared resources without compromising the hypervisor layer. As the work reported in [236, 45, 128], one vulnerability in sharing resource like hypervisor can eventually lead to the compromise of the entire cloud infrastructure and services. For example, a malicious VM can infer some information about other VMs through shared memory [102]. Using covert channels, two VMs can communicate bypassing all the rules defined by the security module of the hypervisor [197].
- *Malicious insiders:* According to the report of Symantec [236], 8% of the reported data breaches in 2014 were the result of insider theft. Also, [141] reports that 70% of attacks on organization sensitive data and resources can be caused by insider attack. Thus it is not surprising to see malicious insiders in the NFV environment as well. A practical example about malicious insiders has been discussed in [128], in which the malicious administrators can take the memory dump of a user's VM. Because they have the root privilege to access the hypervisor, and by using search operation they can extract the user ID, password, and SSH keys from the memory dump. This leads to the violation of user privacy and data confidentiality exposure.
- *Untrustworthy service composition:* One of the salient features of NFV is dynamic service composition, which means the service components provided by different service providers can be composed on demand. However, those service components must be authenticated and the trust relationship between them should be established beforehand. Any vulnerability of service component can ultimately put the whole service at risk. Moreover, the dependencies between different service components are extremely complicated, making the trust relationship twisted [55]. Such complexity holds true for the heterogeneous hardware, software, and the tenants [221].

3.3.2 Use case 2: Virtual Network Platform as a Service (VNPaaS)

Thanks to NFV, the service providers can provide a suite of infrastructure and applications as a platform, on which the enterprises can develop and deploy their own network services and applications that are customized to meet their business purposes. Basically, it allows enterprises to have full administrative control, and can apply all configurations based on their needs with

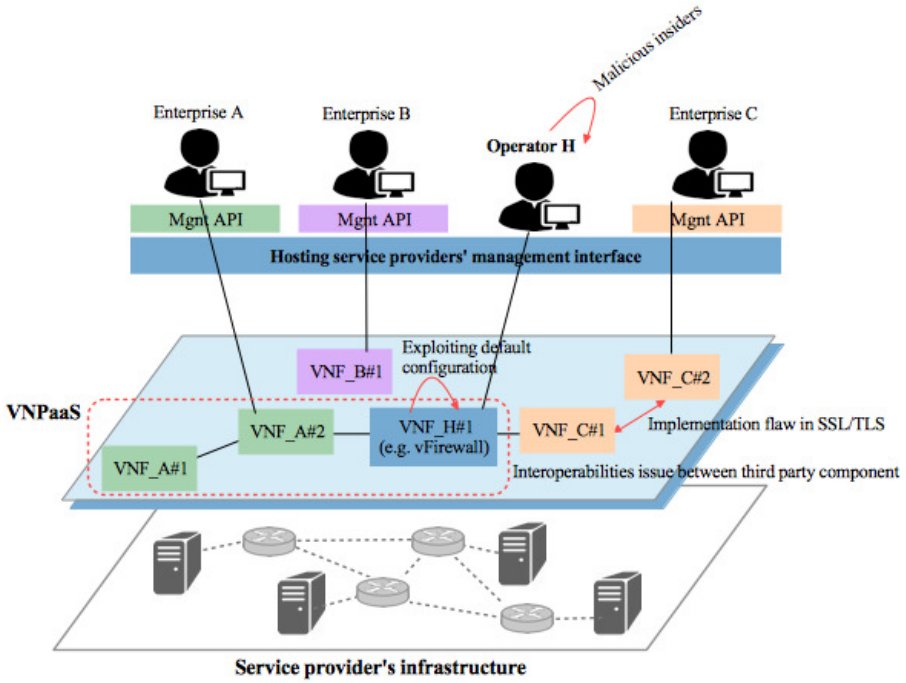


Figure 3.5: VNPaaS use case: an example of sharing network resources, together with attack models

potential support from the hosting service providers. The architecture of VNPaaS with sharing network resources is shown in Fig 3.5, in which the hosting service providers own the infrastructure and rent the shared infrastructure resources to the third parties. Enterprises can deploy either a stand-alone VNF instance that does not have any connection with other VNF instances in the hosting service provider's network, or several VNF instances can be connected to a VNF instance in the hosting service provider's network. To do that, there is an orchestration interface for each entity which is used to specify policy rules and communicate between the VNF and service providers.

However, VNPaaS model gives most of resources and security control to the consumers, while the service providers only preserve the fundamental security of underlying infrastructure. The threat vectors of VNPaaS are therefore significantly multiplied, and some of them are explained as follows.

- *Exploiting default application configurations:* In a VNPaaS scenario, the enterprises usually run their applications and services upon the service provider's infrastructure, which normally rely on a basic set of configurations [49] . This, however, could lead to backdoor attacks. For example, the firewall usually manufactured with default administrator login credential can be easily hacked. If attacker successfully gets in to the firewall, he can modify the configuration rules and let malicious traffic go through.

- *Implementation flaws of SSL/TLS:* In VNPaaS scenario, the authors of [49] pointed out that if SSL/TLS protocol has been selected and used by the hosting service providers to create secure communication between two parties, the flaws and security vulnerabilities resulting from misconfigurations of SSL/TLS have become an attractive target in the hacking community. For example, attacking on Public Key Infrastructure (PKI), handshaking protocol [151, 30]. Also, the pseudo random numbers (nonce) used by SSL/TLS can possibly be predicted and compromised, because it relies on the predicable values such as current time and process id which are vulnerable to get brute force attacks.

- *Security breaches resulting from lack of interoperability:* VNPaaS provides a large set of services and powerful control, allowing enterprises to manage and orchestrate their services on demand. However, the diverse computational resources, multi-vendor software stacks, and the sophisticated isolation of the workloads generated by different enterprises and operators may lead to security breaches, *e.g.*, isolation failure risk, vendor lock-in, and incompatibility with other components. As pointed out in [241], if the platform resources are not accessed and managed with standard solutions, or security mechanisms are not applied to separate workloads in an appropriate way, the quality of service could seriously downgrade even leading to malicious attacks. Also, Takabi et al. [221] indicated that the root causes of interoperability issue is the lack of well-defined standard. Apparently, security threats can potentially result from such standard inconsistency.

- Security flaw in development life cycle: One of the major benefits of PaaS is that it provides a platform allowing enterprises to develop, run, and manage their applications without the complexity of building and maintaining the underlying infrastructure resources. However, from the perspective of application development lifecycle, the developers can be faced with the complexity and inconsistency of building and maintaining secure applications [102, 155]. For example, the frequent updating security patches in the central hosting cloud are definitely affected both system development lifecycle and security flaws in enterprise's applications, if the developer does not take into account about software updates and changes. In other words, the developers have to understand that any changes in PaaS components can compromise the security of their applications.

- *Malicious insiders:* The threat of malicious insiders has been previously identified [128], showing that the lack of transparency of operational processes and procedures in VNPaaS model make malicious insiders difficult to prevent. For example, a malicious system administrator working for hosting service provider can use his authorized user rights to gain access and collect confidential data from his enterprise assets without permission.

Generally, it is worth noting that since VNPaaS use case is based on virtualization layer, security threats and vulnerabilities occurred at virtualization layer (*e.g.*, hyperjacking, VM escape, VM hoping, and VM DoS) could also impact business operations and security posture of VNPaaS use case as well.

3.3.3 Use case 3: Virtual Network Function as a Service (VNFaaS)

In this use case, the enterprise is defined as consumer of the services who is able to configure the applications, without being able to control and manage the underlying infrastructure. The service provider can scale the NFVI resources for specific VNF instances to increase their capabilities, making network services and VNF functionalities available to the enterprises as a service. In fact, the concept of VNFaaS is similar to VNPaaS, but different in the scale of service and programmability, and the scope of control provided to the consumers. Additionally, VNFaaS is only limited to configure the set of VNF instances, so the threats occur in VNFaaS can also appear in VNPaaS. Fig. 3.6, exemplifies the most prevalent transition models of VNFs such as routing, firewall, IDS/IPS, DNS, NAT, and Content Delivery Network (CDN).

As the service providers do not expose internal structures, the users may have limited visibility and controllability over the network resources. These intrinsic limitations lead to significant security threats, which are analyzed as follows.

- *Vulnerabilities in VNF softwares:* In particular, VNF softwares have been used by the network operators for deploying and launching network functions on demand. However, they are likely to be vulnerable for various types of software flaws [199]. For example, the

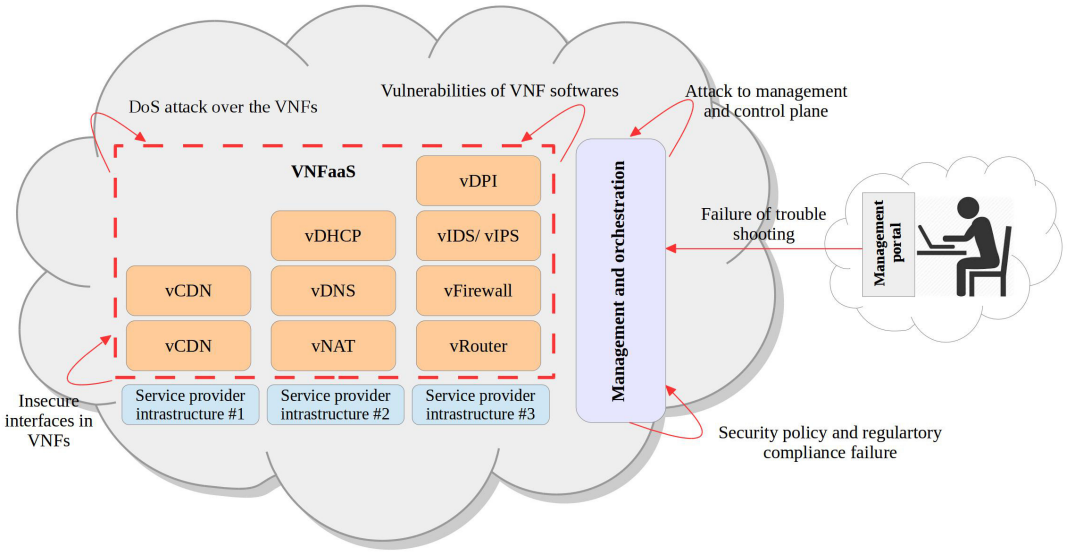


Figure 3.6: The overview of VNFaaS and its attack models

remote attackers can leverage this software flaw to bypass intended firewall restrictions via crafted packets (CVE-2012-2663), or taking advantage of stack-based buffer overflow for allowing them to execute arbitrary code within the victim VNFs like Snort (CVE-2006-5276). Despite these types of software flaw have already been patched, they give a first insight on the possibilities of security issues that threaten network functions when they are deployed in NFV environment.

- *DoS attack:* This kind of attack can either directly target to the VNF instances or VNF's public interface by flooding all available resources with heavy amounts of unreal traffics. The objective of DoS attack is to consume resources such as memory, CPU, and network bandwidth, in an attempt to make them unreachable to the end users or other VNF requests. For example, an attacker may compromise one VNF, by simply generating a large amount of network traffics, then sending out to other VNFs which either running on the same hypervisor or on the different hypervisor. The ultimate goal is to exhaust network resources and impact service availability. Lal et al. [128] depicted one practical scenario of DNS amplification attack which is originally inspired by DoS attack. As shown in Fig 3.7, the service provider's infrastructure hosts a virtual DNS server as a component of a virtual Evolved Packet Core (vEPC). The deployed DNS is dynamically scaled-out when the traffic load increases. The attacker sends IP packets from a false (or spoofed) source IP addresses of victim hosts to launch a large number of malicious queries. Thus the NFV orchestrator will instantiate new VMs to scale-out the vDNS functions to accommodate more queries. In the worse case scenario, if the service provider's infrastructure cannot instantiate more vDNS functions to support dramatic amplification of DNS queries, this situation can lead to service disruption or unavailability. Another example of DoS attack has been discussed in [37]. According to the report, Bitbucket - a web based hosting service company hosted by Amanzon, was attached by massive-scale DDoS attacks used by two flooding techniques, a flood of UDP packets and a flood of TCP SYN connection requests. The attacks caused company's service unavailable, making many developers lost access to the projects hosted on Bitbucket. In addition, the availability of VNFs especially due to attacks like DoS/DDoS needs to be carefully guaranteed in order to ensure the overall quality of NFV service [33].

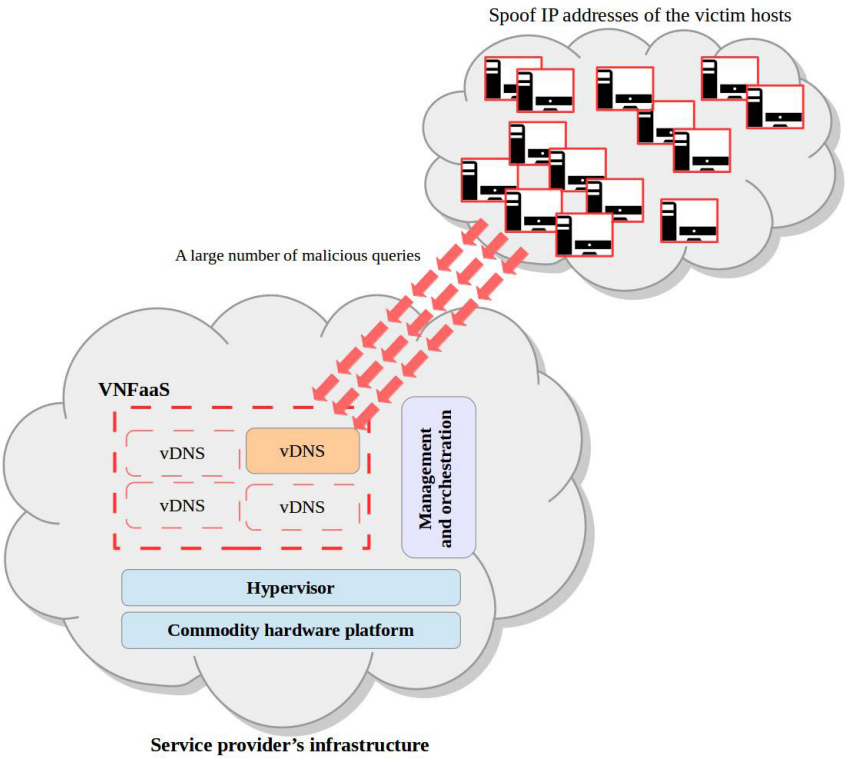


Figure 3.7: An example of DNS amplification attack

- *Attacks to management and control plane:* Basically VNFaaS moves network services and VNF instances to the virtualized environment, so as to allow network operators to manage network assets from a central control point. On this aspect, NFV management and control plane is considered as the brain of all operations, including VNF creation, configuration, management, provision, and monitoring network services. However, the management and control plane could become the single point of failure and attractive attack target, so any compromised operations may lead to the failure of the entire system [254] or getting a wider network impact [52]. For example, if the attackers can gain access to the control point, the entire network services will be under their control. They can freely reconfigure network services, disrupt data path, monitor data transaction, and further acquire the confidential and sensitive information of the victims.

- *Failure of troubleshooting:* NFV management and orchestration needs to correlate information across heterogeneous entities, potentially increasing operational complexity despite the friendly interfaces. Clearly, this makes determining the root causes of service failures challenging. For example, the end-to-end latency of network service could be caused by network service itself, the platform which host virtualized network functions, or any other components across the network. Thus, the interleaved dependency between those functional components makes it extremely difficult to infer the root causes of failures. Chayapathi et al. [34] pointed out that fault detection and troubleshooting in NFV environment are challenging. As a matter of fact, any troubleshooting processes require debugging at multiple levels. A problem arises in one level can impact other levels as well. For example, when a failure of fault occurs at a lower level such as hypervisor crashes, it might generate failures to all VNFs deployed on the top of that hypervisor. Thus debugging at the VNF level alone is not sufficient enough to figure out the root cause of the problem. Also, Lal et al. [128] investigated troubleshooting failure, in which the compro-

mised VNFs can generate a large amount of logs on the hypervisor, making it difficult to analyze the logs from other VNFs.

- *Security policy and regular compliance failure:* In general, the management and orchestration module is used to control the connectivity between VNFs, create relationships between VNFs and virtual infrastructure resources, maintain a sequence of network forwarding path, and address the entire service function chaining processes. However, the lack of consistency on how to orchestrate, manage and operate the network services can incur security threats. Due to the fact that many NFV vendors independently implement VNFs without sufficient collaboration between each other, thus multi-vendor integration make it difficult to coordinate global security policies [52]. Also, the different tenants requires specific security protection and system requirements, this can generate difficulties and create complex trust issues [241]. In addition, Lal et al. [128] exemplified an attack model based on violation of regulatory policies, by moving one VNF from a legal location to another illegal location. The consequences of violating regulatory policies can be in the form a complete banning of service or exerting a financial penalty, which may be the original intension of the attacker to harm the service provider.

- *Insecure interfaces:* The interfaces are widely used by applications to execute network services on virtual infrastructure, or communicate to VNF manager and orchestration system. Considering the prevalence and significance of these interfaces, adequate knowledge about their implementations, deployments, and operations are necessary. Yang et al. [241] pointed out that defining standard interfaces for various security functions is a big challenge when implementing security services in a virtualized network platform. Also, the authors of [236, 45, 128] mentioned that the security and trust concerns must be carefully taken into account in their entire lifecycle, as any implementation errors or misconfiguration can expose the interface to attacks. For example, attackers may exploit the insecure VNF API to dump the records of personal data from the database to violate user privacy, or they may embed malicious codes into the compromised interfaces to escalate their privilege and access to network assets.

3.3.4 Use case 4: Virtualization of Mobile Core Network and Mobile Base Station

As today's mobile networks are populated with a large variety of proprietary hardware appliances, NFV has significant potential to reduce network complexity and cost of ownership, improve QoS, increase data rates and processing capability. Thanks to NFV, it is possible to create innovative implementations of the third party network applications by unlocking the proprietary boundaries of current mobile core network, IP Multimedia Subsystem (IMS), and mobile based station, further consolidating them into virtualization environment. Specifically, virtualization of mobile core network can be applied to Evolve Packet Core (EPC) domain that encompasses all network functions of the mobile packet core, such as Mobility Management Entity (MME), Serving/Packet Gateway (S/P GW), Home Subscriber Server (HSS). For virtualization of mobile based station, it aims at leveraging IT virtualization technology, at least a part of Radio Access Network (RAN) node and Based Station (BS) to standard IT servers and storage. Example functionalities include baseband radio processing unit, Radio Resource Control (RRC), Radio Link Control (RLC), and Packet Data Convergence Protocol (PDCP). Fig 3.8, presents a conceptual idea of virtualizing mobile core network and mobile base station. The benefit of virtualization of mobile network gives operators the ability to seamlessly interoperate the vEPC, vRAN, and vBS with the physical packet core network functions, enabling multiple operators to share the same infrastructure and allowing different operators to control their own network functions. In addition, it is possible to virtualize only a fraction

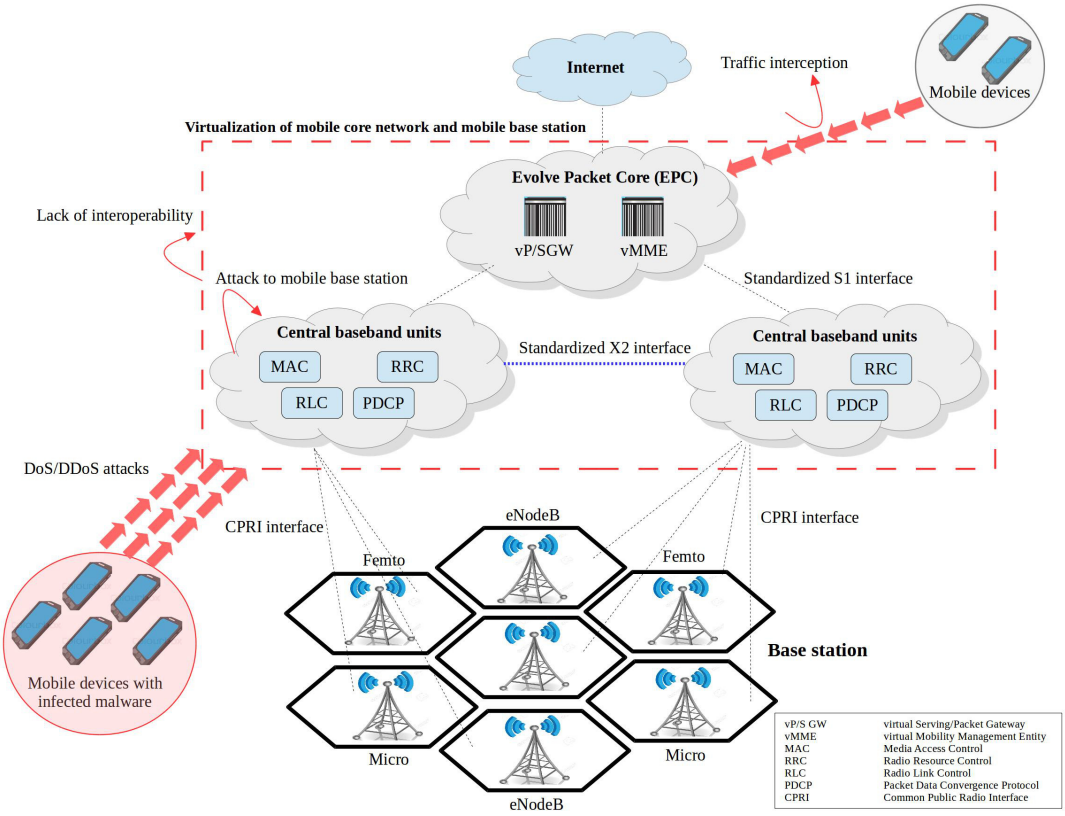


Figure 3.8: The virtualization of mobile core network and mobile based station with attack models

of the components or network functions in the mobile core network and mobile base station, thereby allowing the mobile network operators to deploy a set of virtualized network functions together with the non-virtualized ones.

However, the large increasing deployment of mobile broadband IP networks, resulting in a large variety of security threats and vulnerabilities, which could appear at different levels, *e.g.*, service component level, administration or operation boundaries, between VNF components and NFV infrastructure. The specific threats with regards to virtualization of mobile core network and mobile based station are shown in Fig 3.8.

- *DoS attack to mobile network:* As similar to other use cases, attackers can exploit mobile devices to launch DoS/DDoS attacks to make network services unavailable. Mavoungou et al. [143] pointed out that mobile devices are crucial elements in mobile network security, since they can be targeted and used by the attackers for launching DoS attack towards mobile network. For example, using mobile malware like Trojan horse, a mobile device can turn into a botclient, thus enabling it to receive commands from remote attacker to control or corrupt the victim server. Lal et al. [128] exemplified a practical DoS attack on mobile core network like Mobility Management Entity (MME). To do that, attackers create a botnet army by infecting many mobile devices with a remote-reboot malware, in order to enable them to instruct the malware to reboot all mobile devices at the same time. The simultaneous rebooting of all devices causes excessive malicious attach requests and results in a signaling storm, putting vMME under DDoS attack.

- *Traffic interception:* In general, network traffic between mobile devices are based on IP, and most of mobile devices are still relatively unprotected. Attackers can take this opportunity to sniff and capture legitimate traffic, similar as what they do in the traditional mobile networks. Also, a compromised mobile device can extensively scan other locally

adjacent peers for consuming the limited spectrum [139]. As identified in [143], many types of traffic can be intercepted, including the edge cache traffic and voice calls. For example, the attackers can manipulate the frequently requested content that is stored at the edge of the mobile core network, they can also track and intercept VoLTE traffics or data communication incoming to and outgoing from the Internet.

- *Attacks to mobile based station (e.g., eNodeB, Femtocell, Microcell):* To achieve resource utilization and reduce the operational cost, mobile network operators are adopting virtualization technology for mobile core network and mobile base station. However, some security vulnerabilities still exist. As discussed in [139, 143], a common eNodeB (or 4G based station) uses a virtualized Linux operating system instead of a custom operating system (that has been explicitly hardened and made more secure during thier development lifecycle). If a virtualized eNodeB is successfully attacked through a security flaw in the commercial hypervisor or operating system of application, *e.g.*, radio application software, then the entire processes will get affected. Similarly, if virtual radio access network nodes are compromised, the overall network management infrastructure can be easily controlled by the attackers. Regardless of specific strategies, the attackers can eventually manage to disrupt network services, or cause outages once they have penetrated into the mobile network infrastructure.
- *Lack of interoperability:* Current mobile core network and mobile base station are mainly deployed with physical network functions and purpose-built hardware devices. Not all of them can adapt to virtualization environment, due to the built-in proprietary protocols. As pointed out in [139], RAN nodes are usually implemented on purpose-built hardware, because the baseband processing function can not be efficiently implemented on software. Considering this fact, an alternative solution is to make virtualized and non-virtualized network functions co-exist in the network. However, such heterogeneous mobile network environments can create sophisticated management tasks, and lead to challenging security issues.
- *Insecure content and media delivery:* The number of users of mobile devices continues to grow sharply, unavoidably leading to the increase of mobile attacks in term of number and sophistication. Despite the diversity of attack variants, attackers are always motivated to steal data and user identities. Especially in the virtualization environment, the data is stored in the public cloud infrastructure and frequently accessed by multiple consumers, a variety of threats and attacks including the zero-day ones could possibly occur. Without sufficient data protection mechanisms, user's data can be at risk. As reported in [139], unauthorized access and DoS/DDoS attack on content and media delivery can impact business revenue and degrade the quality of service delivery. Also, Mavoungou et al. [143] pointed out that the emergence of mobile network virtualization has opened a large scale deployment for mobile applications and services, while bringing many security issues such as unauthorized data access, traffic eavesdropping, data modification, and theft of services.

3.3.5 Use case 5: Fixed Access Network Functions Virtualization

It is well known that the major performance bottleneck generally occurs at the network access node. Thanks to NFV, the fixed access network functions can be migrated to virtualization environment, so that multiple organizations can share the resources with a dedicated partition of a virtual access network. Basically, the virtual access network functions are applied initially to hybrid fiber Digital Subscriber Line (DSL) nodes, such as fiber to the cabinet (FTTcab) and fiber to the distribution point (FTTdp). Fig 3.9, illustrates the idea of access network

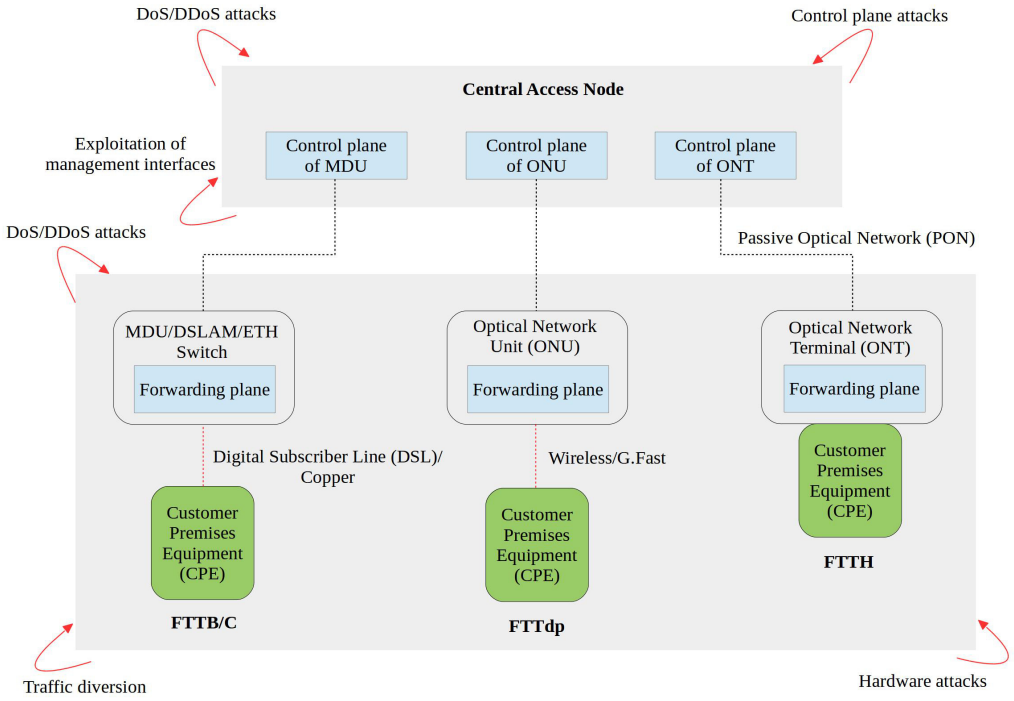


Figure 3.9: The virtualization of access network functions with attack models

virtualization, in which the forwarding plane is separated from the control plane, while the control plane runs in central point and managed by the NFV abstraction layer. The target network functions that can be virtualized, such as Optical Line Termination (OLT), Optical Network Terminal (ONT), Optical Network Unit (ONU), Multi Dwelling Unit (MDU), and Digital Subscriber Line Access Multiplexer (DSLAM).

Although the virtualization of fixed access network functions provide additional benefits to support multi-tenancy environment in term of low cost, power consumption saving, and automated provisioning, security threats and vulnerabilities are potentially introduced, which are discussed as follows.

- *DoS/DDoS attacks:* These attacks are very common threats to cloud and NFV environment including the virtualization of access network functions. It aims at causing reduction or disruption of network services. DoS/DDoS attacks can happen at any layer. For example, at the forwarding plane, attackers may expose a target system to a large number of requests (*e.g.*, a flood of UDP packets or TCP SYN connection requests) in order to overload the available resources, bandwidth, and link capacity, resulting the access nodes are not able to serve other user's requests any longer [199]. At the control plane, DoS/DDoS attacks can be caused by congesting controllers through a large number of forged flow arrivals, causing network performance degradation and interruption [66].

- *Control plane attacks:* The purpose of access network functions virtualization is to share the fiber backhaul, power connection, and associated centralized computational resources by moving control plane to operate and manage at the central point. As such, network operators can easily and remotely configure access network nodes, reducing time and making it flexible to add new network services at a large scale. However, controlling and managing all services at one point in the central office may raise the single point of failure and control plane attacks [52, 254]. Any vulnerabilities and attacks occur at the control plane can affect the entire network services. For example, if the control plane is

compromised, attackers can perform illegal operations, such as disrupting the data path, altering the traffic, reconfiguring the fixed access nodes, or re-updating policy rules.

- *Exploitation of management interfaces:* In general, management interfaces are used by the network operators to remotely interact with access network nodes that are deployed in different geographic locations. However, the way on how to standard interfaces for various security functions and management purposes is a big challenge [241]. It is reasonable to assume that if the management interfaces are not designed with standardization or securely implemented, adversaries can gain unauthorized access to the access network nodes and perform harmful actions, such as shutdown, reboot, or reconfigure the traffic paths [52].

- *Traffic diversion:* This threat aims to compromise a network element by diverting traffic flows in the data plane for eavesdropping purpose. The European Union Agency for Network and Information Security (ENISA) [66] pointed out that traffic diversion occurs when the mandatory isolation between slices (a dedicated partition of a virtual access network for multiple tenants) is compromised, or when the enforcing access control policy for a particular slice has been misconfigured.

- *Hardware attacks:* Generally, hardware design and manufacturing occur prior to software development, so it is extremely important to address hardware security in product lifecycle. Because once the attackers compromise hardware module, software security mechanisms running on this devices will be compromised as well. In [176, 66], the most common types of hardware attacks have been discussed. For example, manufacturing backdoors eavesdropping, inducing faults, and hardware modification tampering through jailbroken software.

3.4 NFV Layer Specific Threat Taxonomy

The finding obtained in the use case driven threat analysis (Section 3.3) allows us to establish a comprehensive threat taxonomy, further providing essential information to help the service providers to gain a holistic understanding on the attack surface, ultimately laying down a foundation to develop and deploy NFV based effective security mechanisms. For better illustration, we refer to the ETSI NFV reference architecture in Fig 2.4, and further conduct NFV layer-specific threat taxonomy. The key results are summarized in Fig 3.10.

As shown in Fig 3.10, the attack surface of NFV is significantly broaden, covering from infrastructure layer (*e.g.*, hypervisor, VM, and hardware vulnerabilities) to VNF and NFV MANO layers (*e.g.*, VNFs and network services vulnerabilities, orchestration and control plane attacks, isolation failures, policy violation, and lack of interoperability). Thanks to the NFV layer-specific threat taxonomy, we observed that a large set of vulnerabilities, especially the ones in NFV infrastructure like VMs and hypervisors are critical, because they are commonly shared by more than one layer. Therefore, we believe that the security threats in NFVI layer, *e.g.*, those identified in use case 1 and use case 2 (Section 3.3), deserve more attention and careful studies than the ones in the upper layers. In other words, the lower layer of security vulnerabilities may affect the upper layers, incurring so-called *cross-layer attacks*. It is worth noting that, however, some threats such as malicious insiders, insecure interfaces, data leakage, DoS/DDoS attacks, and security policy violations, may occur in several layers, but their impacts are not necessarily cross-layered. Meanwhile, some threats such as hypervisor attacks and shared virtual resources, are caused by virtualization layer, due to NFV inherits common attributes and characteristics from virtualization concept, so that many security issues occurred in virtualization can possibly occur in NFV environment as well.

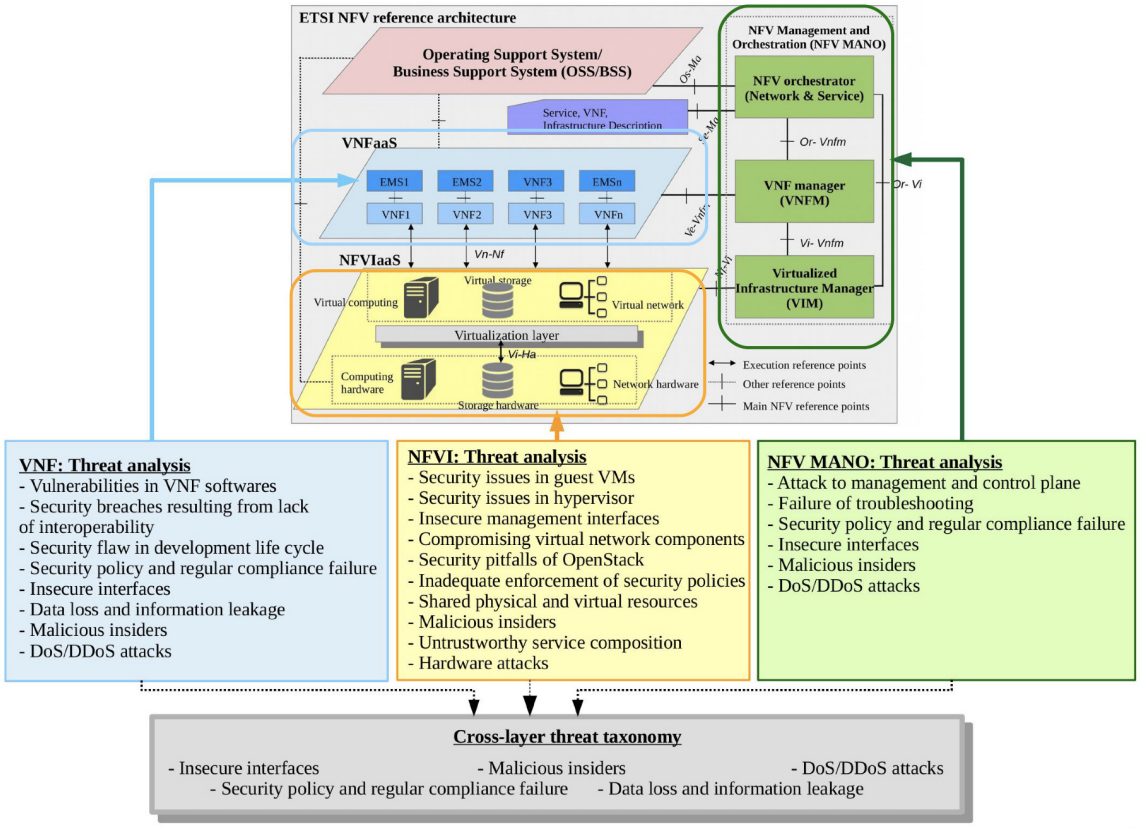


Figure 3.10: NFV layer specific threat taxonomy based on the ETSI NFV reference architectural framework

Another concern is that when considering the large-scale deployment of NFV across different cloud datacenter and security domains, the frequent migration brings a large set of challenges to specify and enforce appropriate security mechanisms and policies. It can be concluded that the basic countermeasures and fundamental security solutions for reducing the amount of attack vectors in NFV environment heavily depend on the sufficient protection of NFVI layer including VMs and hypervisors. Also, the interfaces between VNF instances, the management and orchestration platforms, as well as their service dependencies and consistencies, should be carefully scrutinized to achieve secure and dependable end-to-end service delivery. As a result, the developed comprehensive threat taxonomy serves for two purposes: (1) helping service providers to deploy cost-effective security mechanisms according to their particular contexts or business needs; and (2) potential countermeasures for attack prevention, detection, mitigation, and reaction can be holistically established.

3.5 Security Mechanisms: Comparative Studies

To date, many types of security mechanisms have been applied in practice, such as Identify and Access Management (IAM), Intrusion Detection and Prevention (IDS/IPS), network isolation, and data protection. Although we have seen their successful applications in the traditional network scenarios, it is unclear whether they can be effectively implemented and deployed in the given NFV use cases. Therefore, this section is intended to provide a set of comparative studies on several typical security mechanisms, by firstly analyzing their implementations in IT, telco, and public cloud scenarios, then specifically addressing NFV based implementations. For each security mechanism, a comparative analysis is discussed. Our objective here is to analyze their implicit relationships with NFV performance in terms of feasibility, agility, effectiveness,

and so on. The findings can help the service providers to get a clear understanding about a set of concern metrics when implementing or deploying them in NFV environment.

3.5.1 Identity and Access Management (IAM)

The purpose of IAM is to enable individuals to access the right resources at the right time with the right privileges. It is used to initiate, capture, record, and manage user identities and their related access permissions to information assets in an automated fashion. Thus the access privileges are granted to the users according to the interpretation of policy rules, which are then enforced by a sequence of authentication, authorization, and auditing functions. In addition, the implication of IAM has two independent elements: identity management, and access management. Identity management describes the process of authentication, authorization, and user privileges across system boundaries. Access management is more focused on access control which verifies whether the users are granted privileges to access the requested services/resources. The decision result is evaluated based on policy rules, user's roles, and other elements that are predefined by the administrators.

3.5.1.1 Typical implementations

This section exemplifies the applications of IAM in several typical scenarios to illustrate their implementation and deployment, further analyzing this function based NFV implementation.

- *IT scenario:* One of the typical scenarios of IAM has been discussed in [196], the authors proposed an IAM architecture which relies on two major processes: registration process, and authentication process. In the registration process, all the recipients must be registered to the payment system. Specifically, four good fingerprints from the recipient and recipient's information must be stored in the database. These personal information are also replicated and encoded into the smart cards before issuing them to the recipients. The authentication process is activated when the recipients are required to receive the grant, their smart cards are swiped and the beneficiary place their fingers onto a biometric reader. Then the fingerprints are verified with the fingerprint's information stored in the database and those encoded on the smart card. If the authentication is successful, the recipients can receive the financial grant.

- *Telco scenario:* Location-based access control policies for telco scenario was proposed in [8], by considering both user's location and their privacy. Compared with the conventional access control system, more parties are involved: (1) *requesters*: whose access requests to a service must be authorized by a Location-Based Access Control (LBAC) system; (2) *Access Control Engine (ACE)*: if the evaluation result of access requests match the LBAC policies then the ACE enforces access control to the available services; and (3) *location service*, which provides the location information to the ACE by measuring position and environmental condition of requester.

  However, user privacy in location based service remains to be an important issue [38]. With untrustworthy location service provider, the revealed private location information of requester could be abused by the adversaries. Therefore, location privacy based anonymity solution for the purpose of blinding user's requests and queries was proposed in [222]. The proposed framework is classified into two major parts: (1) authentication process, in which one-way hash function has been applied to provide a better privacy authentication; and (2) querying process, time-fuzzy logic is used to examine the degree of confidence whether the requester is requesting the service under the right privileges, thus both processes are done via anonymity (trusted third party).

- *Public cloud scenario:* As cloud service providers may have different owners and users, this paradigm makes access control and user identity privacy protection is highly complex. Xiong et al. [239] proposed a privacy preserving access management (PRAM) scheme to address identity privacy and access control concerns in cloud services. The PRAM applies both blind signature and hash chains to protect identity privacy and secure authentication, and integrates the demand of access control with SLA to provide flexible fine-grained access management. It adopts attribute-based access control mechanism for authentication process, while the verification is carried out based on the description of user's attributes and SLA. If the authentication is successful, the users are allowed to access the service of interest.

  Another approach proposed in [242], which aims at providing strong and flexible authentication, and data loss prevention. The architecture contains four main components: cloud resource provider, Identity Management (IdM), Policy Management (PM), and Resource Engine and Policy Decision-making (REPD). When the REPD receives the requests, it submits the requests to the IdM to check whether the user is authenticated. If the condition is true, the REPD submits a query to PM to verify whether the user's requests match with the policy rules. Once authorized, the user can access to the requested resources.

3.5.1.2 NFV based implementations

To date, some efforts on virtualizing IAM mechanisms have been proposed, with the purpose to supporting a large number of VNFs running in NFV environment. Some examples are given as follows.

- *Access Control Virtualized Network Function – AC-VNF:* Jacob et al. [115] proposed AC-VNF to control a large number of VNF appliances and authenticate the end users. In particular, the authentication and authorization mechanisms are required to verify the VNF appliances, whether they are eligible to access the requested resources. Also, the service providers who owned the network infrastructure and shared resources can define arbitrary security policies based on their needs, so that each VNF appliance is controlled independently according to those predefined policy rules. The core concept of AC-VNF is implemented based on IEEE 802.1X standard (which is IEEE standard for port-based Network Access Control (PNAC)), and a modified version of the standard to implement the access control per service (instead of per port). More specifically, the AC-VNF architecture consists of four major elements.
  - *End user:* who requests for the services. The machine running the user's request is directly connected to one of physical port of any OpenFlow switches.
  - *SDN/NFV controller:* maintains the state of user traffic. It performs two tasks: (1) redirects user's requests to access control VNF box for authentication; and (2) redirects end users to the requested resources/services, after they are successfully authenticated.
  - *Access control VNF:* which is built on a VM with a Linux distribution, and employs the modified versions of HostAP [106] and WPA supplicant [234]. It acts as software authenticator, implemented based on IEEE 802.1X authentication and authorization (AA). Once the AA procedure succeeds, then the access control VNF box transmits the evaluation result to the SDN/NFV controller.
  - *Service:* which is a specified type of resources provided by cloud service providers. The owner of resources can control them independently using different security procedures.

- *Virtualized access control based on federated identity:* An approach of virtualizing access control based on a federated identity was proposed in [194], with the objective to provide flexible management, increase security, and enable end users to experience seamless single sign-on service. There are two parties involved.
  - *Service provider (SP):* provides the functionality of the application, and controls the access to the resources. It delegates authentication and attribution management to a trusted external identity source.
  - *Identity provider (IdP):* manages the user's identities and their profiles. It adopts a common authentication method and federation standards such as SAML 2.0 [161], OpenID Connect [169], and OAuth 2.0 [100], to manage security and address the complexity.
- *Virtaulization of content-aware identity and access control management:* Another example of virtualizing IAM has been proposed in [235]. The authors aims to build access control system to cover all necessary requirements including privileged user management, fine-grained access control, enhancement of user activity and compliance reporting, sensitive data discovery and information protection. Specifically, three modules are developed:
  - *Role and access policy management:* which pre-specifies a set of different roles of users who want to access the network services, *e.g.*, VMs. The access privileges are granted according to the user's role. In addition, the definition of content-aware identity here is more related to role-based access control.
  - *Virtual systems and applications:* consists of three sub-modules: (1) privilege user management, which controls privileged users using fine-grained access control; (2) compliance reporting, collects user's activities from all event logs; and (3) information protection, facilitates the management of sensitive data.
  - *System and application access:* these applications are running on the VMs, *e.g.*, databases, that have been monitored by security apps.

3.5.1.3 Comparative analysis

To understand the differences between the typical implementations of IAM mechanism and their counterparts tailored to NFV, we take the designs reported in [115, 194, 235] as references for a comparative study. The key findings are summarized in Table 3.1.

- *Flexibility:* In [115], AC-VNF is implemented based on the IEEE 802.1X standard for PNAC, which is a basic NAC solution for enforcing the access control at port level. It is designed to maintain the whole authentication and authorization (AA) processes in the data plane for avoiding overloaded traffic on the control plane. When end user requests to access the services/resources, the requested traffic will be redirected to the access control VNF box, which then authenticates the request based on user authorization rules. The outcome of AA procedure is then transmitted to the SDN/NFV controller. Once the user is authorized, the flow based port entry is generated, thus she/he is allowed to access the requested services. To achieve flexibility, the AC-VNF applied VLAN technique [41] to share one physical port with multiple users. Despite it provides many advantages (*e.g.*, flexibility, cost, and physical layer independence), the scheme has limitations on capacity, security, and overhead. It is worth nothing that VLANs can possibly create 4094 different VLANs for the same network [41]. If a network spans more than one geographical location, the traffic needs to go through the third parties, exposing the traffic to the potential sniffing and man-in-the-middle attack, which are hard to deal with unless higher layers

Table 3.1: The key differences between typical and NFV based implementations of IAM

Scenarios	Example designs	Flexibility	Cost	Complexity of lifecycle management	Effectiveness	Dependability	Scalability
Typical implementations	IAM for social grants in South Africa [196]	✓	✗	✗	✓	✗	✗
	Location based access control policies [8]	✗	✗	✗	✗	✗	✗
	Access control based anonymous location [222]	✓	✗	✗	✓	✗	✗
	Privacy pReserving Access Management for cloud – PRAM [239]	✓	✗	✗	✓	✗	✓
	IAM solution for cloud [242]	✓	✓	✗	✓	✗	✓
NFV based implementations	Access Control Virtualized Network Function – AC-VNF [115]	✗	✓	✓	✗	✗	✓
	Virtualized IAM solution based on federated identity service [194]	✗	✓	✓	✓	✗	✓
	Identity and access management in virtualization environment [235]	–	✓	✓	✓	✗	✓

Notations: '✓' and '✗' denote that NFV characteristics are satisfied and NOT satisfied respectively, '–' means no solid references are available.

offer additional security mechanisms. More importantly, if the VLANs relies on the port based or MAC based configuration, non-trivial effort and time are required to manage the network. An alternative solution is to use VXLAN technology [44], which can be implemented to support flexible and large scale virtualized multi-tenant environment over a shared common physical infrastructure like cloud.

- *Cost:* It is well recognized that NFV can reduce the total cost of hardware acquisition and capital investment through the use of commodity hardware platforms. The report [107] showed that the cost for a small scale NFV deployment can be reduced from $34,015 to $27,828 (about 18%), while a large scale deployment can be reduced from $18,935 to $14,435 (about 24%). From the given examples [115, 194, 235], it clearly identified the advantages of NFV based virtualized IAM solution. The costs from hardware investment and management complexity can be significantly reduced. The access control policy rules are allowed to be redefined and reconfigured remotely in real time. It is worth mentioning that although the cost for hardware investment, installation, configuration, and power consumption decrease, the software cost could increase.

- *Complexity of lifecycle management:* Instead of deploying network equipments at customer sites and using them to provide a set of predefined services, NFV makes it possible to deploy hardware at provider sites and provision services dynamically using centralized management tools. As a result, both cost and complexity with respect to service deployment and configuration can be reduced. Thus security functions like virtualized IAM can be quickly deployed and easily managed through a centralized management platform. As shown in the given examples [115, 194, 235], the virtualized IAM allows network operators to flexibly deploy access control VNF box, define security policies, assign specific actions according to user's attributes and role based access control on the fly. Also, it allows network operators to keep better control over the upcoming user's requests and offer efficient fine-grained policy enforcement.

- *Effectiveness:* If there is a large number of independent VNFs deployed in the NFV infrastructure, the effectiveness of management and communication overhead has become a significant concern. A simple question would be on which location an access control VNF box should be installed, and how many of them should be deployed to achieve the

best performance. Let's consider the AC-VNF architecture in [115], if the end users locate closely to the requesting resources, while the access control VNF box is deployed on a VM which is far from that, then the SDN/NFV controller redirects the network traffic to the access control VNF box regardless considering physical location of the end users. As such, a very large number of traffic over virtual network switches and routers would occur, consuming lots of bandwidth. This clearly indicates that it is necessary to take into account the trade-off between security objective and performance efficiency during the deployment of access control VNF box.

- *Dependability – single point of failure in a centralized controller:* Considering the example design in [115], a controller is deployed to redirect all network traffics to an access control VNF box, and the centralized management can obtain a global view of the network. However, the controller itself could be overloaded, failed, or possibly attacked, resulting in significant performance degradation of the entire network. To solve this issue, a distributed control scheme was proposed in virtualized networks to share the processing load [254]. Thus multiple controller instances are created and organized in a cluster, while the centralized network control is remained. As such, a better trade-off between scalability and dependability can be achieved, as the network control workloads can be distributed across the cluster by deploying each instance of SDN/NFV controller on the dedicated VMs.

- *Scalability:* Theoretically, NFV can make network services agile, cost effective, and scalable. It allows network operators to scale in/out network services on demand, while providing elastic resources management to manage and orchestrate a particular VNF instance when there is a peak traffic load. From the given examples of NFV based implementation [115, 235, 194], scalability is validated. For examples, an access control VNF box can by dynamically activated or deactivated in real time based on the traffic load conditions[115]. The access control policy rules of specific access control box can be dynamically created and modified [235]. The virtual resources such as storages can be elastically provisioned to support a large amount of metadata [194]. As a matter of fact, the access control VNF box usually contains a large number of databases and directories, which are used to store user's identities, attributes, policy rules, and so on. Thanks to NFV, the identities across heterogeneous sources can be easily orchestrated, while providing a logical view, no matter where or how they are stored.

3.5.2 Intrusion Detection and Prevention (IDS/IPS)

In general, IDS/IPS has been designed to protect critical assets against cyber threats by examining system events and network traffic flow of interest. Specifically, IDS is designed to detect vulnerability exploits against a target application or service, so it needs to monitor all the inbound and outbound network activities, and identify whether there are any suspicious patterns or anomalous behaviors existed. The main function of IDS is to warn administrators of suspicious activities rather than taking immediate actions to prevent any malicious activities. Meanwhile, IPS is used to monitor network traffics and identify anomalous activities. It is considered as an inline security component that is able to actively prevent attacks from happening, while IDS is considered as a passive monitoring.

3.5.2.1 Typical implementations

Despite the similar design principles, the design architecture of IDS/IPS vary significantly. In the following, we exemplify the design of IDS/IPS in several typical scenarios.

- *IT scenario:* Jamshed et al. [119] proposed a highly scalable software IDS architecture, called *Kargus*, to detect malicious attack patterns with high speed performance. The core part of Kargus uses modern hardware innovation (*e.g.*, multiple CPU cores), Non Uniform Memory Access (NUMA) architecture, multi-queue 10 Gbps Network Interface Cards (NICs), and heterogeneous processors like Graphic Processing Units (GPUs). Architecturally, Kargus contains three modules: (1) *preprocessing*, which captures incoming packages, reassembles IP package fragments, verifies the checksum of TCP packets, and manages the flow content for each TCP connection; (2) *multi-string pattern matching*, which performs pattern matching by scanning the entire payloads; and (3) *rule option evaluation*, if the signature is matched, the packets are evaluated against a full attack signature relevant to the matched string rules, and ultimately produce the evaluation output.

  Another long standing issue in IDS/IPS domain is to run the system over encrypted traffic like HTTPS. Sherry et al. [211] proposed a Deep Packet Inspection (DPI) middleboxes, called *BlindBox*, to perform packet inspection directly on the encrypted payload without decrypting them. The BlindBox architecture comprises three major components: (1) *DPIEnc* and *BlindBox Detect*, which use searchable encryption scheme and fast detection protocol to inspect encrypted traffics for certain keywords; (2) *obfuscated rule encryption*, essentially relies on the techniques proposed in [243, 193] to enable the middleboxes to obtain the encrypted rules based on the rules from middleboxes and the private key of endpoints, however neither the endpoint nor the middleboxes can learn the rules or the private key; and (3) *probable cause decryption*, which allows flow decryption when a suspicious keyword is observed in the flow.

- *Telco scenario:* As mobile ad hoc networks (MANETs) represent a large class of telecommunication networks, where the nodes share the same physical medium, network topology is dynamic, centralized monitoring and management points are unavailable. These characteristics lead to broad attack surface, ranging from passive eavesdropping to active signal interfering. Therefore, an intrusion detection and response system was then proposed in [250], in which individual IDS agents are placed on each node to monitor malicious activities, and they cooperatively participate in global intrusion detection. Specifically, each IDS agent comprises six modules: local data collection, local detection engine, cooperative detection engine, local response, global response, and secure communication.

  However, the IDS architecture mentioned above does not provide detailed design of cooperation detection algorithms. Thus, Morais et al. [154] proposed a new distributed IDS to exchange events and cooperation between the nodes. The proposed system is based on a popular IDS tool, namely *Bro* [26], which runs on each node to passively monitor the traffics and towards detect attacks in collaboration with its neighbors in real time. Specifically, the proposed architecture is divided into two layers: (1) event engine layer, which transforms a stream of filtered packets into a stream of higher level network events; and (2) policy script interpreter layer, which executes security policy scripts and specifies event handlers.

- *Public cloud scenario:* With the rapid development and deployment of cloud computing, security service vendors are migrating their services to the public clouds, making them as on demand service. Some advantages of cloud based IDS/IPS were discussed in [162], which provides better detection of malicious software, enhances forensic capabilities and retrospective detection, improves deployability and management.

  An alternative approach of cloud based security monitoring was proposed in [153]. This framework integrates a Network Intrusion Detection System (NIDS) in the cloud infrastructure, and uses snort [200] and Decision Tree (DT) classifier [99] to detect network

attacks. It consists of four main components: packet processing, intrusion detection, storage, and alert system. More specifically, the storage component has been designed with three databases: (1) *knowledge database*, stores known attack signatures; (2) *behavior database*, stores network behaviors having both malicious and normal packets; and (3) *central-log database* is used to record malicious event's log that reported by Snort or DT classifier.

Despite the widespread adoption of security monitoring like Snort and Bro, their technical limitations always remain in the typical application scenarios. For example, setting up a small scale Snort instance is a well documented activity, but it quickly becomes a significant engineering challenge when network size gets large. Security monitoring tool like Bro may provide flexibility, customization, and analysis capabilities, but setting up Bro is a complex process that requires domain expertise. To tackle these problems, Shanmugam et al. [209] proposed a distributed elastic intrusion detection architecture, called *DEIDtect*, to provide distributed framework for cross-site intrusion detection. In particular, the DEIDtect exploits cloud computing to consolidate resources to handle the computing need by IDS/IPS tools. It applies SDN/NFV technology to allow administrators to monitor network traffic at any point in the network, and send traffic stream of interest for further anomaly detection.

3.5.2.2 NFV based implementations

This section exemplifies several NFV based IDS/IPS implementations.

- *Security monitoring appliance – CloudSec:* Ibrahim et al. [112] proposed a monitoring appliance, called *CloudSec*, to provide active, transparent, and real time security monitoring. Thanks to Virtual Machine Introspection (VMI) technique [87] that offers fine-grained inspection of VM's physical memory. Thus multiple VMs hosted on a cloud platform can be concurrently monitored without installing any monitoring code inside the hosted VMs. There are two main components at the VMI layer, which serve as the core of CloudSec architecture: (1) Back-end component, which enables the hypervisor to gain control over the hosted VM to suspend any access to physical memory and CPU. It performs necessary security checks and alerts the front-end when malicious attacks occur. (2) Front-end component, is a set of APIs that obtain information about the monitored VMs, and control access to physical memory and CPU register. It is considered as an external extension of hypervisor that enables transparent access to physical memory without installing any additional security code.
- *Encrypted virtual machine introspection – CryptVMI:* Yao et al. [244] proposed an encrypted VMI system, called *CryptVMI*, to provide users a complete status of their virtual instances, while keeping confidentiality of user's data by using encryption technique. The designed architecture of CryptVMI consists of three modules: users, query handler, and introspection application. At the first time of user registration, the user is assigned a unique ID and a random symmetric key. These two parameters are then passed as inputs in encryption process for securing communication between the user and query handler. When the query handler receives user request for anomaly detection, it firstly checks whether the user is associated with any VMs. If so, it uses the cloud service API to locate IP address of the compute node holding that VMs, and then uses introspection application to inspect malicious attacks.
- *Security health monitoring – CloudMonatt:* Another security monitoring solution was proposed in [248], called *CloudMonatt*, which provides a flexible distributed cloud architecture to detect and monitor the security health of customer's VMs based on a rich

set of security properties for VM attestation. It built upon the property based attestation model, and provided several novel features including monitoring different aspects of security health, mapping actual measurement that can be exploited by customers, and taking countermeasures based on the monitored results. Specifically, four modules are developed:

- *Customer:* CloudMonatt gives the customers two modes of operation: (1) one time attestation, which allows customers to request attestation at any time; and (2) periodic attestation, which allows customers to ask for periodic attestation.
- *Cloud controller:* acts as the cloud manager for receiving VM requests and serving them for each customer. Three modules are involved: (1) *policy validation module* is used to collect the monitored security measurements from the VMs of concern in response to the requests of customers; (2) *deployment module*, which allocates each VM on the selected attestation server, the measured result is then report back to the controller; and (3) *response module*, provides appropriate countermeasures if potential vulnerabilities are occurred in the VMs.
- *Attestation server:* is responsible for validating security measurements, interpreting properties, making attestation decisions, and issuing an attestation certificate for the VMs that are monitored. In fact, attestation server is used to monitor malicious attacks on the customer's VMs, while the cloud controller is responsible for management.
- *Cloud server:* the computer that runs multiple VMs.

3.5.2.3 Comparative analysis

It is clear that NFV based IDS/IPS implementations have certain advantages over the typical implementations. This Section is intended to provide in-depth comparison between the typical implementations of IDS/IPS and their NFV based implementations. The key results are summarized in Table 3.2.

Table 3.2: The key differences between typical and NFV based implementations of IDS/IPS

Scenarios	Example designs	Flexibility	Centralized control and management	Complexity of lifecycle management	Specific function	Scalability	Cost
Typical implementations	Highly scalable IDS – Kargus [119]	✗	✗	✗	✗	✓	✗
	Deep packet introspection – BlindBox [211]	✗	✗	✗	✗	✗	✗
	IDS for mobile network [250]	✓	✗	✗	✗	✓	✗
	Distributed IDS for mobile network [154]	✗	✗	✗	✗	✓	✗
	Malware detection – CloudAV [162]	✓	✗	✗	✗	✓	✓
	NIDS for cloud [153]	✗	✗	✗	✗	✓	✓
	Distributed elastic IDS – DEIDtect [209]	✓	✓	✓	✓	✓	✓
NFV based implementations	Security monitoring appliance – CloudSec [112]	✓	✓	✗	✗	✓	✓
	Encrypted virtual machine introspection – CryptVMI [244]	✓	✓	✗	✗	✓	✓
	Security health monitoring – CloudMonatt [248]	✓	✓	✓	✓	✓	✓

Notations: '✓' and '✗' denote that NFV characteristics are satisfied and NOT satisfied respectively.

- *Flexibility:* It has been observed that many high performance of IDS/IPS based implementations rely on dedicated network processor and RAM, this increases the cost and makes operation lack of flexibility. In contrast, NFV based implementations consolidate

security functions onto industry standard platforms located in distribution center. Thanks to the capability of NFV automation and orchestration, the cost and operational complexity can be reduced. Also, NFV aims to increase compatibility with existing deployment and facilitate interoperability. These properties are more or less validated in the aforementioned references. For example, the basic idea of CloudSec monitoring [112] utilizes VMI technique to provide monitoring capability of inspecting security threats over virtual infrastructure, without installing any security code inside the VMs.

- *Centralized control and management:* Traditionally, network operators need to set up and configure dedicated IDS/IPS boxes directly, which usually makes configuration and management process highly complicated and time consuming. The given example like CloudMonatt [248] shown that NFV has potential to reduce the complexities. Thanks to a controller module, a simplify management can be achieved. Based on the design principle of CloudMonatt, the controller firstly accepts the requests from customers to monitor security health of their VMs. It then assigns attestation server to inspect customer's VMs whether there is any potential vulnerabilities occurred. The attestation server sends the result back to the customers after successful verification. In addition, the controller carries out appropriate response once vulnerabilities are found.

- *Complexity of lifecycle management:* Today's IDS/IPS are usually deployed and operated behind the firewall to provide a complementary layer of detecting malicious contents. It is placed in-line (in the direct communication path between source and destination), actively analyzing and taking automated action on all incoming traffic flows. They have been configured and controlled by network operators. When transitioning to virtual IDS/IPS appliances, service providers who provided underlying infrastructure and security services have to deal with a lot of complexity, spanning from installation, configuration, to management and maintenance. Additionally, the complexity of network management on independent security monitoring could make it more complicated to handle and potentially produce communication overhead. These results bring negative impacts to system performance. For examples,

  - *Operational complexity:* The given example like CloudSec [112] utilizes VMI technique to monitor volatile memory for further detecting kernel rootkits. The challenge of implementing CloudSec is how to map the introspected low-level raw bytes of memory into high-level OS data structure instances. To do this, it contains several sequential steps of detecting threats in VM's physical memory. Unavoidably, this process introduces operational complexity in terms of communication overhead and process synchronization.
  - *Management complexity:* In CryptVMI [244], every communication between users and management node need to be encrypted using a random symmetric key, which is initially provided by its management node. When query handler module (in management node) receives user request, it firstly checks whether there is any associated VMs belonging to this users. If the condition is true, the query handler uses cloud service API to locate the IP address of the compute node that holds the designated VMs, and sends inspection request to introspection application to monitor malicious events. In case, there is a large number of users that are concurrently requesting the services, management node could be faced with management complexity and communication overhead, and could lead to single point of failure.

- *Specific functions:* Traditional IDS/IPS is often placed directly behind the firewall to provide a complementary layer for monitoring, identifying, and blocking malicious events.

Having an IDS/IPS in-line means that all the incoming and outgoing traffics of a corporation network will be captured and analyzed. However, the capabilities of IDS/IPS are limited to support monitoring VMs running inside the host, and it is difficult to monitor malicious processes and activities, as the nature of virtualization makes cloud environment complicated. These facts lead to the development of VMI technique [87]. Nevertheless, embedded VMI normally requires customized settings that fit into specific environment, making VMI hard to be integrated with the existing security monitoring tool such as Bro. Moreover, applying VMI technique may break down the boundaries of segregation between multiple tenants, leading in exposure or leakage of data privacy. Although Yao et al. [244] proposed an encrypted virtual machine introspection system – CryptVMI, to maintain user confidentiality, the extra cost of encryption and computation on encrypted data is still incurred.

- *Scalability:* NFV allows service providers to scale up/down network services on demand, significantly improving service request and response time, making them react faster to update configurations, *e.g.*, reconfigure forwarding paths, resetting policy rules. This property has been validated in the aforementioned references. For example, in CloudMonatt [248], the network operators can dynamically add or remove new cloud servers, and reconfigure the desirable security properties in security policies in real time without affecting the overall system performance.
- *Cost:* If an enterprise network gets larger, then hardware investments on IDS/IPS are definitely increased, along with the increasing efforts on configuration and operation. By using NFV based IDS/IPS solution, such as the one reported in [248], the cost and management complexity can be reduced. The customers can leverage a centralized security monitoring to perform checking processes, and inspect suspicious events at the end node without incurring additional capital expenditure and installation cost.

3.5.3 Network isolation

Network isolation is considered as an essential building block for improving security level and ensuring security control in resource sharing and data communication over the cloud environment. The first aspect of network isolation is to physically or logically segment networks to provide secure communication and offer higher bandwidth for specific users. The second aspect is resource control or QoS management, which relies on efficient traffic monitoring and management to ensure that users only consume their share of network bandwidth. Additional mechanisms are also required to improve network traffic isolation, such as SLAs establishment to meet QoS requirement, bottleneck identification to prevent network congestion, and security information gathering to prevent attacks.

3.5.3.1 Typical implementations

The applications of network isolation are exemplified as follows.

- *IT scenario:* Juba et al. [121] proposed security control, called *POSTER*, to keep an intra-LAN in a secure state by isolating those network devices, *e.g.*, computers, network printers. The proposed architecture consists of three components: IDS, OpenFlow switch, and OpenFlow controller. At first, the IDS monitors all the communication inside the LAN to detect attacks or preliminary attacks from LAN, and then the detected information is sent to OpenFlow controller. This controller analyzes the detected information to perceive the current LAN situation, and examines the device where the anomalous communication come from. Flow control is then applied to this device using OpenFlow switch.

Another example of network isolation was discussed in [218]. The authors proposed *Quarantine* model to resolve the problem of network separation deployed across IPv6 applications and services. The network nodes are accommodated to separate network segments according to their security level, while different security policies are applied for specific network segments. Based on the Quarantine model, there are the Quarantine server acting as a monitoring server to monitor security level of node, and policy enforcer which accommodates the node to a network segment based on its security level.

- *Telco scenario:* Baliga et al. [11] proposed a Virtual Private Mobile Network (VPMN) to improve network isolation over a mobile wireless network. The idea of VPMN is aimed to leverage virtualization benefits in order to dynamically create private network and provide resource isolation on a shared mobile infrastructure. Especially, there is a VPMN controller which is responsible for accepting user requests, creating, and manipulating infrastructure for that user. Thanks to the benefits of virtualization and partition mechanisms provided by VPMN, so that each user equipment has its own mobility elements (*e.g.*, virtual network service instances for RAN/eNodeB, S/P GW), instead of sharing network elements like traditional model.

  Another example is *TrustDroid* framework - a practical and lightweight domain isolation on Android [28]. It aims to mitigate unauthorized data access and provide secure communication between applications using different trust levels. The basic idea of TrustDroid is to group applications in isolated domains, while the applications belong to different domains are not permitted to communicate among each other. Specifically, the isolation is applied for three layers of the Android software stack: (1) at the middleware layer, which prevents inter-domain application and data access; (2) at the kernel layer, by enforcing mandatory access control; and (3) at the network layer, which mediates network traffics based on policy setting.

- *Public cloud scenario:* Brassil et al. [25] proposed to use Coarse Wavelength Division Multiplexing (CWDM) optical network technology to explore the strategic use of heterogeneous physical network hardware for isolating different tenants in the clouds. The main idea of CWDM is to multiplex multiple independent communication channels on a single optical fiber using passive optical devices. As a result, one single fiber can carry multiple sub-channels which are segregated based on wavelength and the associated color-coding.

  Other techniques of network isolation in the cloud has been discussed in [44, 43]. Traditional mechanism like VLANs suffer from limited scalability and inefficient use of available network links (with a maximum 4094 VLANs) [41]. As a result, Virtual Extensible LAN (VXLAN) solution was proposed to provide elastic workload placement, higher scalability of layer 2 segmentation, and large scale flexibility to support multi-tenant cloud over a shared physical infrastructure. In fact, the original objective of VXLAN is to provide the same Ethernet layer 2 network services as VLAN does, but with greater extensibility and flexibility.

  In [231], the vShield solution was proposed to have a more efficient security model. It offers customers with different trust levels through the use of complete network isolation, so that all the application deployments are enforced using the proper segmentation and trust zones. Several key services are provided by vShield. For examples; (1) *vShield App*, which creates application boundaries based on policy enforcement; (2) *vShield Edge*, which maintains secure multi-tenancy by employing load balancing and isolation services, *e.g.*, firewall, VPN, NAT; (3) *vShield Endpoint*, providing on-host anti-virus and malware protection; (4) *vShield Manager*, a central point of control is introduced to manage, de-

ploy, and orchestrate third party security services; and (5) *vShield Zones*, which provides basic protection in virtual network such as traffic analysis.

3.5.3.2 NFV based implementation

As previously discussed, a large variety of isolation techniques were proposed in the past decades. However, when the size of cloud network gets large, virtualizing network isolation become a promising approach. This Section is intended to exemplify how network isolation can be virtualized in the context of NFV.

- *Virtualized network isolation – DCPortals:* Nunes et al. [160] proposed a virtual network abstraction, called *DCPortals*, to provide traffic isolation in a virtualized datacenter environment. It offers logical isolation among multiple tenants that share the same infrastructure, without requiring any additional hardware. Thus each tenant's resource is isolated from others, and the tenant can see his VMs as connected to a single virtual switch. The core concept of DCPortals was implemented as a network hypervisor module built on top of POX controller [183]. It interacts with OpenStack for querying tenant's information, and interacts with Open vSwitch (OVS) to identify network isolation based on OpenStack's RSA access key. The DCPortals assume that all VMs that shared the same RSA key belong to the same isolated network. If two VMs are found in separated networks, a packet sending from one VM to another VM is dropped by the first OVS to avoid extra processing in the network. Otherwise, both VMs are considered to be in the same virtual network, thus the traffic from one can reach the another.

- *Trusted Virtual Domains – TVDs:* Berger et al. [17] proposed Trusted Virtual Datacenter (TVDc) to provide strong resource isolation and integrity guarantee. In TVDc, a group of VMs and resources that collaborate among each other, called *Trusted Virtual Domains (TVDs)*. The TVDs maintain strong isolation between workloads by enforcing a Mandatory Access Control (MAC) policy throughout the datacenter. This policy defines which VMs can access to which resources, and which VMs can communicate with each other. It ensures that resources allocated to one TVDs can not be made accessible to VMs of another TVDs. According to the design principle, the isolation is performed at two levels.

  - *Hypervisor based isolation:* The TVDc is implemented based on hypervisor security architecture, called *sHype* [203], with the purpose to enforce information flow, and control access to VMs and resources based on security policy. It used MAC to provide access control, thus the requested communication to the VMs and resources is permitted or denied according to the access control policy rules.
  - *Network based isolation:* The TVDc applied VLANs to create isolated network, in which a single physical LAN is segmented with appropriate VLAN ID. Thus all VMs in TVDc are allowed to connect with other VMs when they are holding the same VLAN ID. In other words, two VMs with the same VLAN ID can communicate freely through VLAN, while the communication is denied if their VLAN ID are not matched.

- *Logical isolated network partitions:* The authors of [40] proposed path isolation mechanism to maintain end-to-end isolated network virtualization. Thanks to virtualization, logical isolated network partitions can be created on the top of physical network infrastructure of an enterprise, while providing the same services as those in traditional enterprise networks. The concept of path isolation can be classified into two categories.

– *Policy based path isolation:* which restricts the forwarding traffic to a specific destination based on a policy and additional information provided by the forwarding control plane.

– *Control plane based path isolation:* which limits the propagation of routing information – only the node that belongs to the same subnet can communicate among each other. Virtual Routing and Forwarding (VRF) [42] can be used to achieve this approach, which allows customers to virtualize a network device from a layer 3 standpoint and further creates different virtual routers in the shared physical device.

3.5.3.3 Comparative analysis

In this Section, we give a comparative analysis, mainly addressing the gap between typical based and NFV based implementations of network isolation. To do that, we refer to the aforementioned examples as references [160, 17, 40]. Finally, the key results are highlighted in Table 3.3.

Table 3.3: The key differences between typical and NFV based implementations of network isolation

Scenarios	Example designs	Cost and efficiency	Scalability	Effectiveness	Centralized control and management	Complexity of lifecycle management	Security enhancement
Typical implementations	Secure intra-LAN with isolation – POSTER [121]	✓	✓	✗	✗	✓	–
	Separated network segmentation – Quarantine [218]	✗	✗	✗	✗	✗	–
	Virtual private mobile network – VPMN [11]	✓	✓	✓	✓	✓	✓
	Lightweight domain isolation – TrustDroid [28]	✗	✗	✓	✗	✗	✓
	Coarse Wavelength Division Multiplexing – CWDM [25]	✓	✓	✓	✗	✗	✓
	Virtual Extensible LAN – VXLAN [44, 43]	✓	✓	✓	✗	✗	✓
	vShield solution [231]	✓	✓	✓	✓	✓	✓
NFV based implementations	Virtualized network isolation – DCPortals [160]	✓	✓	✓	✓	✓	✓
	Trusted Virtual Domains – TVDs [17]	✓	✓	✓	✓	✓	✗
	Logical isolated network partitions [40]	✓	✓	✓	✓	✓	✓

Notations: '✓' and '✗' denote that NFV characteristics are satisfied and NOT satisfied respectively, '–' no solid references are available.

- *Cost and efficiency:* Traditional management model of network services always cause inefficient resource utilization, as specific hardware is assigned for specific function, and resource pools are customized per application usage. By using NFV, capital and operational costs can greatly reduce, as network service delivery is removed from a physical device to a virtual context, without the need to deploy specialized hardware for every network service instances. Also, it can reduce the total cost of ownership by simply sharing each hardware platform among multiple network services. These benefits have already been validated in [160, 17, 40]. For examples, network isolation technique [160] uses virtual switch to isolate tenant's network traffic from others, path isolation [40] employs Virtual Routing and Forwarding (VRF) to create different virtual routers in the same physical devices, and Trusted Virtual Datacenter (TVDc) [17] applies policy rule setting to isolate tenant's workloads from other tenants.

- *Scalability:* Scalability is another important feature offered by NFV. As previously discussed, network service instances and virtual resources can dynamically scaled on the

fly to support the growing demands and dynamically changes according to customer requirements. The following examples have illustrated how scalability and flexibility can be achieved through the virtualized network isolation.

  - *Scalability in service deployment:* According to [40], virtualized network isolation allows network operators to quickly create independent logical traffic paths in a shared physical network infrastructure. As a result, the traffic of one tenant is isolated and invisible to other tenants that do not belong to the same domain.
  - *Scalability in service management:* In TVDs [17], moving from managing the classical individual VM isolation to workload isolation substantially simplifies security management. One major advantage is that network operators no longer need to monitor individual VMs and resources but they can focus on workloads as a whole. Each workload is enforced based on specific access control policy rules. This allows network operators to have a greater flexibility to define the policy rules for specific workloads in a virtualized datacenter. More importantly, network operators can simply create, deploy, and update security policies at the central point in real time without affecting the overall service performance.

- *Effectiveness:* Referring to [17], the concept of TVDc has been proposed for both hypervisor based isolation and network based isolation. Especially network based isolation, the TVDc isolates VMs that contain the common security policies into the same group, then each VM can only access to the resources of other VMs within a group with regards to the predefined security policy rules. Technically, VLANs technique is applied to achieve this goal, by using the VLAN ID to isolate the VM's traffic in the network, limiting one VM to only communicate with other VMs that belong to the same VLAN ID (TVDc group). As such, network isolation can significantly increase the levels of trust, yielding stronger security protection over the shared physical and virtual resources, *e.g.*, VMs.

- *Centralized control and management:* There is no centralized control and management available in traditional network isolation, so that network operators are required to take several steps to configure network isolation (*e.g.*, configuring network interface and other parameters with respect to network isolation), resulting in non-trivial administrative burden. On the contrary, NFV offers centralized control and management to mitigate the complexity of device configuration, while providing an easier way to update policy rules in real time. This benefit has been validated in [160], thereby utilizing a centralized controller the network operators can quickly isolate network traffic flow of interest based on the predefined security policy rules.

- *Complexity of lifecycle management:* Commonly, the configuration of network isolation can be simplified by assigning a dedicated (and possibly overlapped) IP address space, and then segment them into specific subnet associated to the different user groups. However, this approach does not work well for large network environments like cloud and NFV, due to sophisticated routing management. In [160, 17, 40], the authors proposed to leverage virtualization to simplify network isolation. The major idea is that each partition is logically isolated from others by defining a set of policies. This allows the network operators to easily create and modify the isolated virtual networks, while supporting a large number of user groups that have different user requirements. Because security policies are centrally enforced, adding or removing users and services to/from virtual network does not require significant policy reconfiguration.

- *Security enhancement:* VLANs have been widely applied to create logical isolated network partitions over a shared physical network infrastructure. The related example can

be seen in [17]. However, S. Rouiller [201] pointed out that although VLANs make it possible to isolate traffic at layer 2 that share the same switch, the heavy dependence on software configuration makes it vulnerable. In fact, VLANs mainly use to separate subnets and implement security zones. The possibility of sending packets across different zones would make such separation useless, as a compromised machine in a lower security zone could initiate DoS/DDoS attacks against machines in a higher security zone. Another threat is to destroy virtual architecture by launching DoS/DDoS attacks against the whole network architecture, which can significantly impact the business operations.

3.5.4 Data protection

It is widely accepted that data protection plays a vital role in preserving the privacy of personal sensitive information. It gives users with more security and provides a greater control over their data. Fundamentally, data protection involves several types of security functions including data encryption, data isolation, data leakage prevention, and key management.

3.5.4.1 Typical implementations

The applications related to a set of data protection are exemplified as follows.

- *IT scenario:* Chiang et al. [36] proposed secure key exchange protocol, called a Three-Way Key Exchange and Agreement Protocol (TW-KEAP), which enables two communication parties to share a session key for establishing secure communication over an insecure network. It is based on the concept of Diffie-Hellman key exchange protocol that allows the key exchange without session key appearing in the message. There are three parties involved in the TW-KEAP architecture: (1) SIP proxy servers perform key exchange process to derive the session key; (2) gateway is used to forward data stream and assisted the service provider to collect the monitored data stream for lawful interception; and (3) clients who registered under the SIP proxy server intending to create secure communication based on TW-KEAP protocol.

  Other examples of data protection are typically applied in the cloud environment to support secure resource sharing. To do this, there are two feasible solutions of data protection: data isolation at application level and at hardware level. The related example of data isolation at application level was discussed in [118]. The authors proposed isolation system, called *Solitude*, to limit the effects of attacks and simplify the post-intrusion recovery processes. Solitude leverages fine-grained access control for those untrusted applications, and uses isolation to separate each application from others. Also, to restrict attack propagation, it limits system capabilities with least privilege principle when untrusted applications is attempted to request access to the file. For data isolation at hardware level, Strongly Isolated Computing Environment (SICE) was proposed in [10], which is a framework that provides a hardware level isolated execution environment for x86 hardware platforms. In particular, SICE uses security manager to prevent the isolated workloads from accessing memory of the legacy host. Even a malicious workload in one isolated environment can compromise its own security manager, it is not be able to compromise any other isolated environments running on the same platform.

- *Telco scenario:* Zhong et al. [252] proposed a distributed k-anonymity protocol to tackle the problem of user privacy in mobile Location-Based Services (LBS) from revealing their location or credential to the untrusted service providers. The distributed k-anonymity protocol is based on homomorphic encryption and k-anonymity technique, which allows users

to remain anonymous within a set of k users under a coverage area. The architecture consists of four components: (1) location broker, which keeps track of user's current location in the coverage area; (2) user, who carries mobile devices for getting the current location; (3) secure comparison server, which interacts with the users and let them learn whether there are at least k users who registered to the same coverage area; and (4) directory server, which publishes contact information to both location broker and the secure comparison servers.

Another example was proposed in [89], with the objective to tackle the challenge of key management in mobile network, where the user's locations are frequently changed. The authors proposed a decentralized architecture for group key management to support secure communication among members in the mobile network. Specifically, it provides key generation, dynamically updating the keys (when members move, join, and leave the group), and distributing the keys to the group members. The main idea of this protocol is that multicast traffic is encrypted by the source and decrypted by valid receiver using Traffic Encryption Key (TEK). The TEK is automatically updated when a new member joins the group, or when an existing member leaves the group. The design architecture of decentralized key management is organized into multiple areas. Each of which is managed by the Area Key Distributor (AKD) which shares with its area members a Key Encryption Key (KEK). Such multiple areas that belong to the same cluster are controlled by one Domain Key Distributor (DKD), as its role aims to create a new TEK and distribute this TEK to all the members within its cluster.

- *Public cloud scenario:* Takabi et al. [220] proposed a privacy aware access control system to support secure data sharing in the cloud. It provides two levels of data protection when user's data is stored in the cloud service provider (CSP). At the first level, the CSP applies access control mechanism to protect user's data from unauthorized access. At the second level, it relies on trusted third party service provider to prevent against colluding CSP. To do that, it applies multiple layers of commutative encryption technique [53], so that the users are not required to trust neither the CSP nor the third party service provider. The data owners can protect their resources from untrusted CSP using encryption, and from unauthorized users using fine-grained access control policies.

  Another approach of data protection in cloud has been proposed in [240]. The authors applied Ciphertext Policy Attribute-based Encryption (CP-ABE) to achieve fine-grained access control. In particular, the owner firstly divides his/her data into several components, and then encrypts each component with different keys by using symmetric key encryption. Consequently, the owner applies CP-ABE to encrypt each symmetric key, such that only the user whose attributes satisfy the access policy in the ciphertext can decrypt the keys.

3.5.4.2 NFV based implementation

In this Section, we exemplify several NFV based data protection mechanisms.

- *Data leakage detection – CloudSafetyNet:* Priebe et al. [184] proposed a monitoring data leakage system, called *CloudSafetyNet*, to examine the data flow of tenant applications in the cloud platform whether there is any data leakage. It adds security tags to a subset of all tenant HTTP request fields, and uses socket-level tag monitoring to observe the propagation of tags. Each monitor maintains a tag log which is periodically retrieved by the tenant. If the tags that belong to other tenants are observed in the tag log, it indicates that there is inter-tenant data leakage.
- *Data confidentiality protection:* In [245], the authors attempt to address the same security issue of protecting data confidentiality when they are stored and processed in the cloud.

A new architecture was proposed to protect user privacy and ensure that service providers are not able to collect any user's data, while the data is processed and stored in the cloud. In particular, three major operations are involved: (1) separating software service provider from infrastructure service provider, so as to ensure that the software layer and infrastructure layer are not managed by the same party; (2) hiding information about the owner's data by using identity anonymization function; and (3) performing data obfuscation when data is stored in the cloud.

- *Key-insulated symmetric key cryptography:* Dodis et al. [62] proposed key-insulated symmetric key cryptography to tackle a problem of cryptographic key compromise. This scheme aims to mitigate the repeated exposure of secret keys, in which the keys need to be updated frequently. The design is based on key-insulated public key cryptography, and integrated into Trusted Virtual Domains (TVDs). In particular, TVDs allow customers to use multiple VMs, only applications in the same TVDs can communicate with each other using key-insulated symmetric key cryptography. Assuming that there are two VMs intend to communicate with each other, they initially need to establish a shared key. The process of generating the shared key is based on key-insulated symmetric key cryptography. As both VMs receive partial secret key from its device master key, so that each VM takes its device master key together with its computer master key as inputs, in order to generate a period-secret key as a shared key. This key is then used to establish secure communication between the two VMs that belong to the same TVDs.

- *Key management in the clouds – EnCloud:* Song et al. [216] proposed encrypted cloud system, called *EnCloud*, to handle the challenge of key management in the cloud, in which a large number of keys need to be properly managed to support multi-tenant requests, and providing end-to-end encryption between multiple cloud applications. The design solves a set of problems of insecure key creation, key management, and key renewal. In EnCloud, all encryption and decryption keys are located at the client site rather that the server site, enabling users to have full control over the use of keys and their personal data management. Specifically, EnCloud consists of two functional components: (1) Domain Manager (DM), which is responsible for managing its members (*e.g.*, domain clients), and issues the domain key; and (2) Domain Client (DC), which is responsible for data encryption and decryption.

3.5.4.3 Comparative analysis

This Section is intended to examine NFV based implementations of data protection mechanisms. We analyze their performance and compare with their traditional counterparts. The findings are summarized in Table 3.4.

- *Cost saving:* One of the major drawbacks caused by traditional data protection is that many enterprises need to rely on dedicated encryption and key management systems to protect their data. However, these types of equipments are very expensive, complex to configure and manage. With NFV based implementation, security services like data protection can be quickly deployed and managed by using software based approaches, avoiding significant investment on hardware. In fact, the aforementioned references [184, 245, 62, 216] have clearly indicated that end users and enterprises can dynamically deploy virtual appliance of data protection service on the fly, without too much concern about costs in hardware investment, configuration, and maintenance.

- *Communication overhead:* Many researchers have pointed out that conventional encryption model may not be sufficient enough and no longer suitable to fulfill the security

Table 3.4: The key differences between typical and NFV based implementations of data protection

Scenarios	Example designs	Cost saving	Communication overhead	Complexity of lifecycle management	Easy deployment (no specific function required)	Effectiveness	Scalability
Typical implementations	Secure key exchange protocol – TW-KEAP [36]	✗	✗	✗	✗	✓	✗
	Data protection at application level – Solitude [118]	✗	✓	✗	✗	✓	✗
	Data protection at hardware level – SICE [10]	✗	✓	✗	✗	✓	✗
	Homomorphic encryption for location privacy [252]	✗	✗	✗	✗	✓	✗
	Decentralized group key management [89]	✗	✗	✗	✓	✓	✓
	Secure data sharing in cloud [220]	✓	✗	✗	✓	✓	✓
	Data protection in cloud using CP-ABE [240]	✗	✗	✗	✓	✓	✓
NFV based implementations	Data leakage detection – CloudSafetyNet [184]	✓	✗	✗	✗	✓	✓
	Data confidentiality protection [245]	✓	✗	✗	✗	✓	✓
	Key-insulated symmetric key cryptography [62]	✓	✗	✗	✓	✓	✓
	Key management– EnCloud [216]	✓	✗	✗	✓	✓	✓

Notations: '✓' and '✗' denote that NFV characteristics are satisfied and NOT satisfied respectively.

requirements in the cloud. For example, users traditionally encrypt their data with secret keys, and then deliver these keys to the authorized user for decryption purpose. This approach makes users a cumbersome process and creates high risk of key exposure. Although the authors of [62, 216] proposed a new model of encryption and key management for cloud environment, in which the end users do no need to manage the keys by themselves, the communication overhead is unavoidably incurred from key exchange. The major reason is that the secret keys need to be updated frequently at boosting period, *e.g.*, when time expired, when the number of communication users changed, or when a device that stored the secret key is lost or stolen. All these cases consume non-trivial computational time, memory, bandwidth, and other resources.

- *Complexity of lifecycle management:* One of the major advantages of NFV is to provide network functions with automatic provision and orchestration to reduce the complexity of VNF lifecycle. However, implementation of data protection in NFV can potentially incur some complexities.

  – *Management complexity:* Unlike the conventional data protection, when data is migrated and processed in the cloud, the users may lose control over their data. Also, service providers may pose a risk to their users by performing unauthorized disclosure of user's data and violating their privacy. As a result, the complexity of data protection could be high, as it requires a complex level of protection to ensure that at every stages of data processing are not disclosed to any untrusted third party except the authorized ones. These goals are partially achieved by the design reported in [62, 216]. Despite the related examples improve robustness of security protection and key management, the management complexities are unavoidable. Especially when considering the processes of key generation, management, and distribution across multiple parties in the large scale networks that having a huge number of users who are simultaneously requesting the keys.

  - *Operational complexity:* Although the design architecture proposed in [184] enhances the confidence of tenants because the data is well isolated and protected, the operations of CloudSafetyNet could increase data traffic in the network, unavoidably producing operational overhead and latency. The reasons is that each HTTP request's packet needs to add security tag, perform encryption, and record log events, as the tenants need to periodically check for data leakage from their log information. Similarly, data protection framework proposed in [245] introduces non-trivial operational complexity. Especially, the separation between software layer and infrastructure layer incur extra communication traffic and data processing overhead between the components belonging to different service providers. The out-of-band operation and synchronization is also necessary.

- *Specific designs:* As previously analyzed, the traditional model of data protection has limited capability to protect data in the virtualized environment like cloud and NFV environments. One of the fundamental weaknesses is that most of them target at preventing data leakage against outsider attacks, while the attacks caused by the insiders such as untrusted cloud service providers can be hardly prevented. Therefore, data protection schemes for virtualization environment are significantly different from the traditional counterparts, as they need to consider the intrinsic characteristics, and the fundamental security requirements of different users. With the objectives in mind, the authors of [245] proposed that service layer and infrastructure layer should not be managed by the same party. Also, the design proposed in [184], every HTTP request needs to add a security tag for observing data propagation and further detecting data leakage.

- *Effectiveness:* In traditional models, the functions of data protection are implemented on specific equipments. For example, enterprise usually relies on a key management device to generate secret keys, and uses an encryption device to encrypt data with those keys derived from the key management device. However, this approach is not effective for virtualization environment that involves multiple networking functions provided by different service providers, while all of them cannot be trusted. A single malicious party can lead to failure of the whole data protection scheme. To deal with these challenges, an alternative solution is to design architecture as discussed in [245, 216], in which users are not required to trust or relied on the service providers.

- *Scalability:* As previously mentioned, NFV provides good scalability to allows service providers to quickly scale in/out network services according to the user demands. One good example is demonstrated in [245], in which service providers can dynamically initiate and deploy software services upon user requests, rapidly interacting with infrastructure layer to allocate virtualized resources, and freely adding or modifying security policy and attestation rules in real time.

3.6 State-of-the-art Security Countermeasures

In this section, we intend to analyze the stat-of-the-art security countermeasures and then propose a set of security recommendations that should take into account to accomplish higher levels of security protection. Thanks to NFV layer-specific threat taxonomy (Section 3.4) and comparative analysis of security function (Section 3.5), giving us a clear understanding to what extent the available NFV security best practices can fulfills these security recommendations. Finally, a summary of NFV's threats and vulnerabilities with the corresponding recommendations is given in Table 3.5.

3.6.1 NFV Infrastructure layer

To defense and mitigate against security threats in NFVI layer, a list of corresponding security recommendations is given as follows. The available security best practices that are tailored to secure NFV infrastructure are also exemplified.

- *Defense in depth with well defined policy enforcement:* To ensure that NFVI layer is properly operated without any attack interruption while maintaining security, defense in depth solution is required. An alternative solution is proposed in [156], the authors classified threat defense solution into four phases of the lifecycle: (1) asset identification phase, which is about identifying users, cloud providers, data, and hardware assets running in NFV environment; (2) adversaries identification phase, by identifying the possibilities of adversaries that could be occurred; (3) layers of defense phase, with an objective to provide security solutions that can be used to tackle the different attacks; and (4) review phase, periodically and frequently rechecking the procedure when user requirements are changed or when security services are out of date.

- *Establishing trust domain:* The concept of trust domain is widely applied in NFV environment to provide confinement boundaries, create secure communication, and maintain security among group's members. One of the basic design assumptions is that each VM needs to trust each other based on a common security policy before starting their communications. In other words, establishing trust between the virtual components is considered as a promising concept to improve security in virtualization platforms. It also helps to maintain the integrity of network. Some examples with regards to trust domain implementations were discussed in [17, 16, 229, 198]. Specifically, several security mechanisms are involved for trust establishment. For example, in the VMM layer, Sailer et al. [203] proposed hypervisor security architecture - *sHype*, to controls resource sharing among VMs based on the formal security policies. In network layer, VLANs [41] and VXLAN [44] are the most widely used techniques for separating VM traffics and isolating them into specific security zone. Thus one VM can only communicate with other VMs that belong to the same security zone. Lal et al. [128] stated out that security zoning can help network operators to prevent a VM from impacting other VMs or host. It is an effective strategy for reducing many types of risks, especially when considering the permeability of existing networks.

- *Dynamic and adaptive access control:* Data stored in datacenter storage across NFVI layer is considered as confidential data and needs to be closely monitored. Like the purpose of traditional access control, access control is used to verify whether the data is accessed by authorized users and know who access the data. As a result, access control models and policy management must be carefully designed in fine-grained, dynamic, and adaptive fashion. For example, the authors of [178, 180] proposed security extension based on access control model, with the objective to provide monitoring, control resource access, and enforce access control policies to protect the deployed resources at NFV infrastructure.

- *Hypervisor introspection:* One solution to defend against attack in VMs is to deploy IDS on the host OS in order to monitor and intercept events from the VMs. This solution is known as *hypervisor introspection* or *Virtual Machine Introspection (VMI)* [128]. The concept behind the VIM is used to scrutinize software running inside the VMs, and then analyze suspicious events and anomalous activities. This technique is widely accepted as a secure model for monitoring the system. It acts as a host-based IDS to monitor network traffics and access to the states of all VMs, such as access files in storage, and read memory execution. Hypervisor introspection can be considered as a powerful tool to perform

deep VM analysis and increase VM security. Also, it can be used as an exploit which make it possible to break down the boundaries of isolation between VMs and hypervisor. A practical example of hypervisor introspection was discussed in [87]. The authors developed *Livewire* prototype to explore potential attacks including kernel-and-user-level rootkits and backdoors, Trojan horses, packet sniffers, and worms.

- *Separation of administrative duties:* When the network functions are migrated to the virtualization environment, the entire network services are controlled and managed directly by the administrators. Assuming that administrators are malicious, the network services can be completely taken over. Therefore, it is important to identify the administrative roles based on their functionalities, duties, and their access rights. Pék et al. [181] identified that administrative roles should be separated, as best practices suggest. As a large virtual environment is supervised by multiple administrators with different roles and duties, so that their access rights should be restricted. For example, a storage administrator should not get access to firewall or monitoring services, and vice versa. Multi-authentication and split-control administrative passwords for various administrators should be used in order to reduce the risks from malicious supervisor.

- *Security service chaining:* In general, the virtual infrastructure should be protected as a whole. As it still needs to maintain open connections for serving network services to external users, additional security services are required, *e.g.*, firewalls, Intrusion Detection and Prevention (IDP), Data Distribution Service (DDS). However, some security services have to work with others to maintain closed-loop security protection or to achieve autonomic controllability. Thanks to NFV MANO, various security services can be initiated on demand. The deployed security appliances can be spun up automatically and chained together for a particular purpose. For example, to defense against DDoS attack, the authors of [46] exemplified a simple model of security service chaining. In this scenario, three types of security services (DDS, firewall, and DPI) are deployed in the different VMs. Based on centralized control and dynamic reconfiguration provided by NFV MANO, all inbound traffics destined for the network service is firstly redirected to DDS, firewall, and DPI, respectively. The goal is to provide optimal efficacy of security service chaining by filtering out malicious DDoS traffic before reaching the targeted network node.

- *Regular VM updates and patches:* To reduce vulnerabilities and mitigate security risks from VM attacks (*e.g.*, hyperjacking, VM escape, VM hopping, VM DoS, VM based rootkits), it is important to keep regular VM updates and patches. As identified in [181, 128], one of the security best practices is to keep the hypervisor up to date by regularly applying the released security patches. The latest security patches should be applied not only to hypervisor and the hosted VMs, but also all the relevant softwares and workloads that are deployed under the NFVI layer.

- *Remote attestation:* Remote attestation is a procedure by which the user is able to verify whether a queried cloud platform is booted in a trusted manner [78]. Also, it can be used to remotely verify the trust status of an NFV platform [128]. The main concept is based on boot integrity measurement by leveraging Trusted Platform Module (TPM). The TPM is used as hardware root of trust for providing trust on a hosting service platform. It is considered as a micro-controller that is often embedded onto the motherboard of services or PCs. This module is capable of storing confidential data (*e.g.*, keys, certificates) and verifying the integrity of system. In fact, TPM is often used during the platform boot time to achieve platform trust. However, remote attestation can be provided as a service at run time before launching other VNF instances [198]. In this scenario, external attestation server is needed to prove the trust status of a remote platform. A practical implementation

of remote attestation service (in form of remote attestation server) is known as open Cloud Integrity Technology (openCIT)[1].

- *Design of trustworthy hardware and platform:* As previously mentioned, hardware design and manufacture always occur before software development, so that the compromised hardware module can make software security mechanisms running upon it become in vain. Therefore, it is important to consider hardware security in the early stage of lifecycle management. For example, Bloom et al. [21] proposed security solution to improve trustworthiness and defend against hardware attacks. The idea is to provide a secure execution environment in the processor supply chain and the hardware mechanism. Also, Lal et al. [128] discussed about Linux kernel security to improve security in virtualization (sVirt). The core concept is based on Security-Enhanced Linux (SELinux). It integrates mandatory access control security with Linux based hypervisor to provide system resource isolation. Thus, data files, network resources, memory, processes, and applications of each VM are isolated.

3.6.2 Virtualized Network Functions (VNF) layer

The threat analysis of VNFs implies that the following security recommendations must be carefully considered. Despite the fact that some security recommendations may also hold true for NFVI, the security countermeasures might be different.

- *Defining standards to support service interoperability:* As a general trend, today's enterprises are migrating their services to the cloud infrastructure. However, without industry-wide cloud standards, vendors tend to create proprietary cloud service based on their unique software stacks. This result leads to service lock-in and incompatibility with that of others. It is therefore important to define standard to support interoperability between the various NFV elements, covering both standalone VNF instances, NFVI platforms, and network functions. In fact, a well recognized community such as OPNFV [174], and OpenStack [170], are some ongoing projects which attempt to achieve this goal.

- *Ensuring secure and dependable of service function chaining:* An end-to-end service delivery in NFV context can be constituted/composed by a series of independent service functions (Firewall, NAT, TCP optimizer, *etc.*) and this is called Service Function Chaining (SFC) in NFV's terminology. Thus, the compromise of any involved network functions in the service chain can definitely break down the consistency of the whole chain. Therefore, it is important to ensure that all specified service functions are chained in a consistent and reliable way, meeting with the predefined high-level SFC specifications. Recently, the authors of [149] proposed a SFC orchestrator to enhance resilience in NFV. The proposed orchestrator activates a standby VNF to replace the failed one, and then employ the SDN controller to redirect the network traffic to the new service function path. However, there are still many open research issues related to SFC that need to be addressed urgently. The most important issue is how to ensure that the consistency of a particular service chain is well protected.

- *Establishing trust relationships between intra, inter, and extra domains:* As mentioned in [76], all parties in the same trust domain can securely communicate with each other. To do that, all VNF instances need to be evaluated for the trust relationship. The evaluation process are based on the contracts, policy rules, and guidelines. However, the trust relationship is broken if one of the VNF instances is not trustworthy. In general, the trust relationship can be defined for three levels: (1) intra-VNF trust, the trust within VNFs; (2)

[1] https://github.com/opencit/opencit

inter-VNF trust, the trust between VNFs; and (3) extra-VNF trust, the trust with external VNFs.

- *Security monitoring:* As in the cloud, the status of infrastructure resources, network services, VNF instances, and other related events are hardly observable. Therefore, the effective security monitoring is needed, that can help to provide visibility into the network events and activities of interest, while enabling the advance logging and accounting function for further forensics and anomaly detection. For example, Zhang et al. [248] proposed CloudMonatt architecture to monitor different aspects of VM's security health. In [150], *Monitoring-as-a-Service (MaaS)* was developed. The idea is to deploy network state monitoring at different levels in the cloud. Also, Zou et al. [253] proposed a trusted monitoring framework for cloud platforms, to provide a chain of trust by deploying independent guest domain for monitoring purpose, and utilizing the trusted computing technology to ensure integrity of the monitoring environment.

- *Data encryption and leakage prevention:* In order to protect data in transit, at rest, and in use, appropriate encryption mechanisms should be applied. For example, Lat el al. [128] identified that virtual volume disks and VM swapping areas associated with VNFs may contain sensitive data, and they need to be protected. One of the best practices to secure the VNF volumes is to encrypt them and store the cryptographic keys at a safe location like TPM module. Also, TLS and IPsec with the latest approved version are recommended for improving secure communication between VNF components [79]. In addition, more advanced encryption mechanisms such as homomorphic encryption [129] can be considered, especially in NFV environment that supports both multi-tenant software applications and service orchestration.

- *VNF image signing and software integrity protection:* VNF software images are used as reference to initiate the VNF instances, so it can be tampered easily by the attackers, *e.g.*, infecting a VNF image file using malware. An alternative countermeasure that was proposed in [128] is to cryptographically sign VNF images, which are then verified when they are launched. However, a signing authority needs to be set up, and the hypervisor configuration needs to be modified in order to verify an image's signature before its execution. Also, as ETSI suggested [79], all VNF instances and related versions of software components must be verified before their execution, in order to enhance software integrity protection. If the VNFs and software components comply with the specified policies, then they are allowed to install. Otherwise, the installation is rejected.

3.6.3 NFV MANO layer

NFV MANO layer plays a vital role in the whole NFV architecture. It acts as a brain of managing and orchestrating virtual resources and network services. The threat analysis has clearly indicated in the previous section. In this section, we intend to identify security recommendations to defend and mitigate against the potential security threats in NFV MANO layer.

- *Security management, orchestration, and automation:* Thanks to NFV that provides additional set of management and orchestration functions, various VNF instances can be instantiated, connected, and chained together to create an end-to-end network service. However, these VNFs require appropriate security mechanisms to be automated, on demand, agile, and friendly interfaced, so that they can be quickly deployed at various security Policy Enforcement Points (PEPs), either across platforms or in the same NFV environment. In particular, the major purpose of security management, orchestration, and automation is to ensure secure and fast service delivery, while improving end user experience. Jaeger

et al. [116] proposed a security orchestrator to manage security functions (*e.g.*, firewall, security zone, etc.). However, no concrete use case and implementation are given. A conceptual design framework of security management and orchestration, called SecMANO, was proposed [178]. The key idea of SecMANO is to introduce a security orchestrator alongside with the conventional NFV orchestrator. These two components thus constitute a new secured NFV orchestrator. A use case of access control is exemplified to illustrate how SecMANO works. Also, reference [180] proposed a TOSCA based security extension for NFV orchestrator by employing a well-developed security policy engine (Moon framework that is developed for access control) [173]. The proposed security extension can automatically verify security attributes specified in the extended TOSCA data model, generate corresponding access control policies, and enforce them to protect the resources in NFV infrastructure.

- *Transparency to network control and management:* Like any other layers, NFV MANO requires confidentiality assurance when data is processed over untrusted cloud infrastructure. In other words, the data should remain confidential and can only operate with the user permission. Since the users differ in privileges, data access must be individualized and restricted to those authorized users. In particular, data protection in untrusted environment normally can be achieved through the following schemes.

  - *Security policy enforcement:* One of NFV benefits is to enable the service providers to enforce centralized policies, to improve control over traffic flow, network programmability, and payload elasticity. Examples associated with policy enforcement were discussed [92, 140], mainly focusing on defining policies to control and manage the behavior of diverse VNF instances.
  - *Authentication and authorization:* Basically, authentication and authorization are widely used to verify the user's identity, prove their privileges before gaining access to the target resources, and then their permissions are granted accordingly. For example, role-based access control [124], attribute-based access control [240], or federated access control [202], can be candidate mechanisms.
  - *End-to-end Encryption:* To protect data confidentiality and integrity, encryption mechanism is always applied to the data that are outsourced to the untrusted network. However, in multi-tenant environment like NFV, an additional key management layer is always necessary for generating, managing, distributing, and revoking the cryptographic keys. For example, a key-insulated symmetric key cryptography scheme was proposed [62], to resistant operations from compromising the cryptographic keys in cloud environment. Song et al. [216] proposed Encrypted Cloud framework (EnCloud), to protect user's privacy and provide end-to-end encryption between cloud applications. Also, to ensure that only authorized applications/devices can decrypt the data on the cloud storage.
  - *Security monitoring and forensics:* According to ESTI standard [80, 78], security monitoring is one of the basic requirements of the network service management systems. The monitoring scope covers network traffics, VNFs, resource usage, system integrity, logs, and other activities which pose security threats. The information gathered from the monitoring can be used for forensics For example, ANASTACIA project [81, 60] developed security monitoring module as a part of security orchestrator plane for collecting security-focused real-time information related to the system behavior from physical/virtual appliances. In this activity, the monitoring data is filtered and aggregated in order to carry out its analysis and the detection of anomalies. It sends alerts to the reaction module in case something is misbehaving.

- *Ensuring controller availability:* As discussed in [52], the controller is considered as the centralized decision point, attacking to the controller would have broad impact on the entire network. Therefore, accessing to the controller should be tightly controlled and monitored. To do that, authentication, authorization, and accounting (AAA) mechanisms are required, and they should be applied at least two layers: the *network infrastructure* to identify the service providers and the *network function layer* to identify the actual consumers. As such, only authorized VNF instances provided by the authenticated service providers are allowed to be executed.

Table 3.5: A summary of NFV threats and vulnerabilities with the corresponding security countermeasures

NFV specific layers	**Threats and vulnerabilities**	**Security countermeasures**					
		IAM	IDS&IPS	Network isolation	Data protection	Security management and orchestration	Policy enforcement
NFVI	Security issues in guest VMs	✓	✓	✓	✓		✓
	Security issues in hypervisor	✓	✓		✓		
	Insecure management interfaces	✓					
	Compromising virtual network components		✓	✓			✓
	Security pitfalls of OpenStack	✓			✓	✓	✓
	Inadequate enforcement of security policies						✓
	Shared physical and virtual resources			✓	✓		
	Malicious insiders	✓			✓		✓
	Untrustworthy service composition	✓				✓	✓
	Hardware attacks	✓			✓		
VNFs	Vulnerabilities in VNF softwares	✓			✓		✓
	Security breaches resulting from lack of interoperability	✓	✓	✓	✓	✓	✓
	Security flaw in development life cycle				✓		✓
	Security policy and regular compliance failure						✓
	Insecure interfaces	✓					
	Data loss and information leakage	✓		✓	✓		
	Malicious insiders	✓			✓		✓
	DoS/DDoS attacks		✓				✓
NFV MANO	Attack to management and control plane	✓				✓	✓
	Failure of troubleshooting					✓	
	Security policy and regular compliance failure						✓
	Insecure interfaces	✓					
	Malicious insiders	✓			✓		✓
	DoS/DDoS attacks		✓				✓

To summarize, although a large set of security mechanisms are readily available, the intrinsic characteristics of NFV bring several challenges to vendors and network operators in terms of design, implementation, and deployment. For example, the design of security functions, the guarantee of network performance, dynamic instantiation and migration, and their efficient placement. Despite the current efforts that address NFV security, a gap with the given security recommendations clearly remains. As a result, we end up proposing a generic framework of security and trust for NFV architecture. As shown in Fig. 3.11, the proposed framework aims to highlight all of the possible security recommendations, expanding both the vertical and horizontal axis, to achieve cost-effective, cross-layer, and in-depth defensive line. It should be noted that some of them are operated in a specific layer, so-called *specific-layer security recommendations*. For example, to defense and mitigate against the security threats at NFVI layer, trustworthy hardwares and platforms, security zoning, hypervisor introspection, regular VM updates and patches, need to be applied. While, some of them are expected to be lunched more than one layers, so-called *cross-layer security recommendations*. These recommendations include security monitoring, identity and access management, trust relationship establishment, policy enforcement, and data protection. However, we admit that, apparently, the framework needs to be significantly enriched with the continuous efforts from the community.

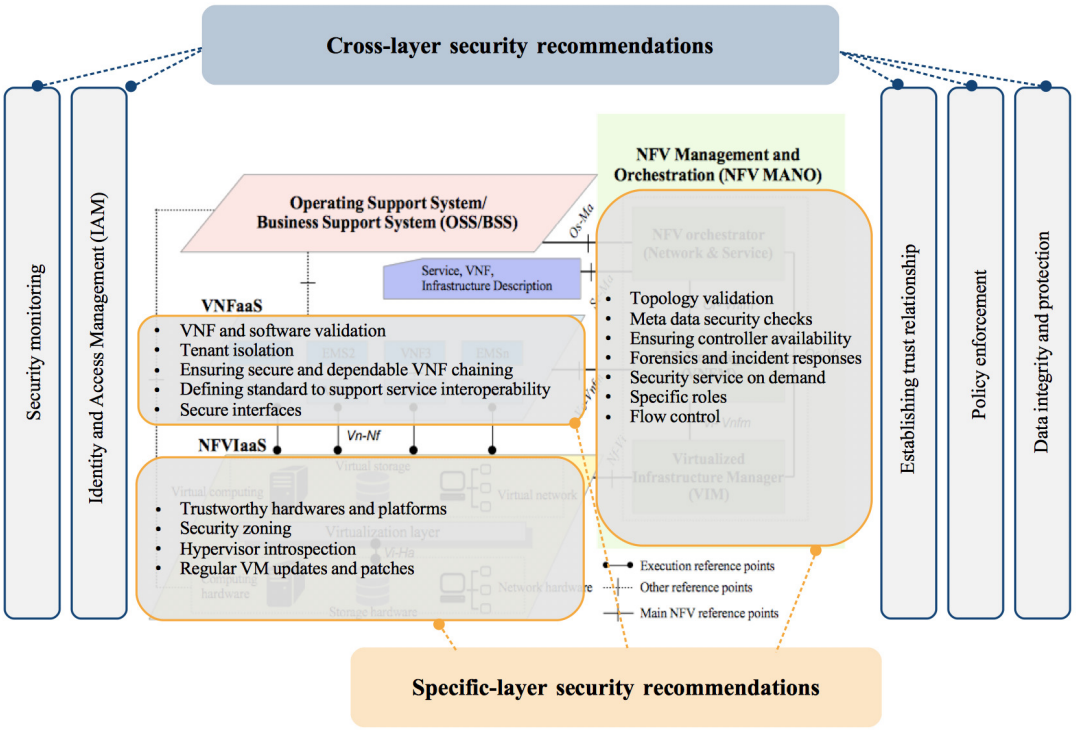


Figure 3.11: A high level of NFV security framework which covers both specific-layer and cross-layer security recommendations

3.7 Conclusion

Recent years have seen the rapid development and deployment of NFV in both enterprise networks and cloud environments. One of the major reasons for this technological trend is that NFV has potential capability of reducing capital and operational expenditures, greatly improving network service agility, flexibility and scalability, as well as resource utilization. As said, any new technologies are essentially two-side sword, while NFV and SDN are not exceptions. Their benefits do not naturally exclude security concerns, which impede the further development and deployment of network services in NFV environment. We are therefore motivated to investigate and explore all potential threat sources associated to NFV.

In this Chapter, we firstly conducted a use case driven threat analysis, in order to establish a NFV layer specific threat taxonomy. In use case-specific analysis, we found that a large set of vulnerabilities, especially the one in NFVI is commonly shared by more than one use cases. For example, if the NFVI layer fails to provide sufficient security protection, those associated network service components in the upper layer would be vulnerable to attacks. This deserves a holistic and systematic analysis of security threats and vulnerabilities across different layers. We assume that the amount of attack vectors can be significantly reduced if the core functional modules of NFV such as VMs, hypervisors, orchestration platforms, and VNF instances are sufficiently protected.

Second, a suit of security mechanisms were analyzed, primarily focusing the feasibility, agility and effectiveness of implementing security functions (*e.g.*, IAM, IDS/IPS, network isolation, and data protection) in the context of NFV. Specifically, we selected five basic security mechanisms and investigated their feasibilities and effectivenesses of NFV based implementations. We found that migrating these basic security mechanisms to the NFV environments can help service providers to save cost, greatly improving flexibility and agility to scale network services up or down to meet dynamically business changing demands. Thanks to the central-

ized management and orchestration, both operational cost and complexity resulting from service deployment, configuration and management can be significantly reduced, making security functions deployed quickly and managed easily. However, we also observed that not all of them can be simply implemented in the context of NFV. For example, as discussed in Section 3.5, evaluating or measuring the performance impact of data protection in terms of communication overhead and complexity of lifecycle management on VNF instances and network services is a non-trivial issue.

Third, based on the established threat taxonomy and the analysis of traditional security mechanisms, the state-of-the-art security countermeasures have been extensively studied. Our objective was to provide a set of recommendations with cost-effective security mechanisms for NFV specific layer, highlighting what threats can be effectively mitigated or prevented by what countermeasures. As a result, we could establish a holistic NFV security framework to present all the possible security recommendations for service providers to achieve cost-effective, cross-layer, and in-depth defensive line. In state-of-the-art security countermeasures, we provided security recommendations for NFV specific layer. We found that an independent security function can not fulfil all the security requirements. In other words, to address a single security recommendation, several types of security functions could be required. Thanks to state-of-the-art NFV security countermeasure, a holistic NFV security framework with cost-effective security mechanisms can be potentially established.

The key observations of this Chapter and those of Chapter 2 lay down a solid foundation for us to develop a security orchestrator, as a natural extension of existing NFV MANO framework, to achieve the capability of managing security functions. More details will be reported in next Chapter.

4 Security Orchestrator for Achieving Software-Defined Access Control

In Chapter 2, we have analyzed NFV architecture and several NFV MANO frameworks (*e.g.*, OpenMANO, Cloudify, Tacker, OpenBaton), and proposed a conceptual framework SecMANO for managing and orchestrating security functions, ensuring all the desirable security properties of network services to be satisfied in their entire lifecycles. This Chapter will be focused on the development of a security orchestrator, which can be viewed as an actual implementation of SecMANO. Specifically, we extend TOSCA data model (our analysis indicates that it has potential to be widely used for today's NFV orchestrators) to incorporate security attributes of interest, and leverage the extended model to create access control policies. Then the design architecture of security orchestrator will be discussed, which contains a TOSCA-parser and a novel tenant-specific access control paradigm. We will demonstrate that our security orchestrator allows to dynamically generate access control models and policies for different tenant domains, ultimately resulting in a flexible and scalable protection across different layers and multiple cloud data centers. Both Proof of Concept (PoC) validation and real-life experiments will be reported for performance evaluation purpose.

4.1 Introduction

Naturally, in order to coordinate the underlying NFV infrastructure resources, initiate a set of VNF instances, and chain them together for delivering end-to-end network service, a functional block of management and orchestration must be put in place. For example, in the ETSI NFV reference architectural framework [73], the core part is NFV management and orchestration (NFV MANO) [74] which plays a significant role in maintaining the full lifecycle management of infrastructure resources, VNFs and network services, ranging from service initiation and configuration, resource orchestration, scaling in/out, to update and termination, to policy management and performance measurement.

Although a number of NFV MANO frameworks appeared and are under development (OpenMANO [223], OpenBaton [168], Cloudify [47], Tacker [219], *etc.*), a majority of them is focused on the basic NFV orchestration and the lifecycle management of infrastructure resources, VNFs, and network services, while security concerns are comparatively overlooked which can be a significant barrier to the wide adoption of NFV [178, 179, 84, 75, 93]. In particular, it remains unclear how the capability of managing security functions can be attained in those NFV MANO frameworks, ensuring that all the NFV-based objects or assets can be protected with the most appropriate security mechanisms in the course of their entire lifecycles, from creation and deployment to the termination.

One of the ideal solutions is to develop a security orchestrator, which can be seamlessly integrated with the NFV orchestrator, enabling the basic security functions to be effectively orchestrated and provided as on-demand services to the customers, and high-level security policies to be specified and enforced in a dynamic and flexible way. Thus, the so called *security by design* has potential to be extensively, if not completely, achieved. To do that, the initial yet essential step is to study the design principle, built-in functional components, and data models of the NFV orchestrators. In particular, a common data model (service template) plays a vital role in managing and orchestrating end-to-end network services by maintaining a consistent view of application topology, network connection, and workloads. TOSCA (Topology and Orchestration Specification for Cloud Applications) [227], for instance, has been widely recognized as a standard approach to defining NFV specific data models and service templates, which are stored in on-boarding catalog and used by the orchestrator for network service instantiation at runtime. To the best of our knowledge, however, the specification of security attributes with regards to virtual components (*e.g.*, VMs, VNFs) remain difficult in the deployment phase, making it extremely challenging to accurately enforce high-level security policies and achieve fine-grained security control in the rest of their lifecycles. Thus, a security orchestrator serves for two functional purposes: (1) closely interacting with the NFV orchestrator to extract security attributes of interest and generate security policies based on the holistic overview of the infrastructure; (2) allowing to dynamically specify high-level security policies and make them enforced in real time to protect NFV resources at both infrastructure and NFV layers, *e.g.*, isolating VNF instances based on tenant-specific domain. In this chapter, we need to achieve the two objectives, so as to eventually develop and implement a TOSCA-based security orchestrator.

Other challenges arise in the NFV infrastructure deployed in the multi-cloud environments, which are essentially multiple tenancy and geographically distributed. In particular, the processes of NFV enabled service compositions may involve different layers (*e.g.*, NFV infrastructure, VNF, and NFV MANO), making service dependency extremely complicated [55, 221]. The security mechanisms deployed in such environments therefore must be sufficiently automated, flexible, adaptive, and scalable. For instance, the access control model and policy need to be tenant-specific, considering the facts that the tenants may vary in security requirements and they may dynamically provision the resources from different cloud infrastructures. In other words, the scope of an access control model and its associated policies should be limited to an individual tenant, while the tenant user is given the privilege to customize the most appropriate access control model and specify the corresponding policies. To do that, we envision that the security orchestrator has a distributed access control framework, which further allows the policy of the tenant to be enforced at the multiple cloud infrastructures.

4.2 Contributions

To address the aforementioned challenges, we have made the following contributions.

- First, we conduct a comparative analysis on the existing NFV orchestration frameworks, especially the open-source ones, with an objective to analyzing (1) whether or not the existing frameworks of NFV orchestration allow to specify, define and manage security attributes in the entire lifecycle of the NFV services; (2) how the security attributes are leveraged and formally specified and modeled as security policies that can be effectively enforced.

- Second, We develop a security orchestrator, which consists of a TOSCA-parser and a software-defined tenant-specific access control engine. The access control engine enables

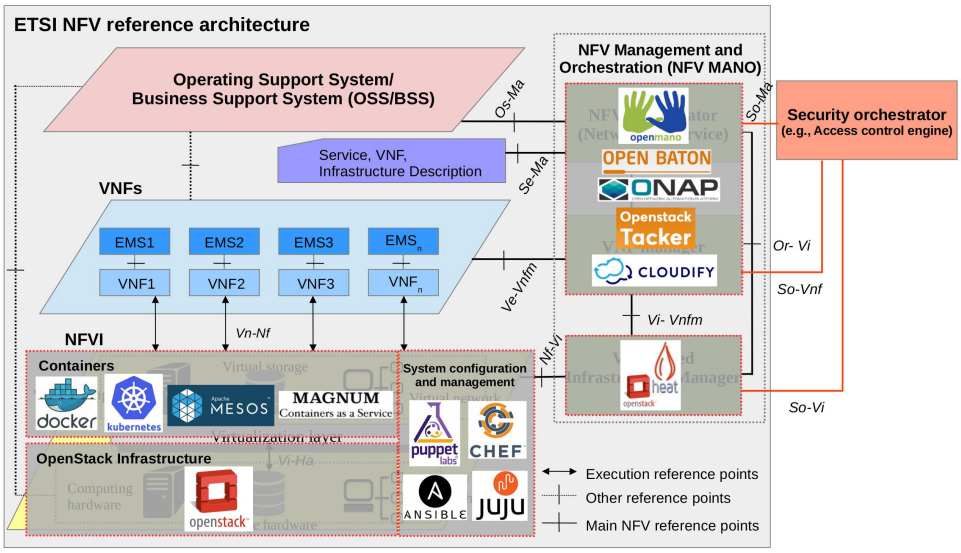


Figure 4.1: The mapping of security orchestrator in ETSI NFV MANO architectural framework

the access control model and policy to be dynamically generated for different tenant domains, which may use virtual resources (*e.g.*, VMs, VNFs) in different geographically distributed data centers.

- Third, We develop a prototype and run a set of experiments in real-world platform and evaluate its performance in terms of throughput, scalability, and adaptability. The experimental results demonstrate that all the desirable properties can be achieved at a satisfactory level regardless of the varying number of tenants, users, or objects that are deployed in the cloud. More importantly, we demonstrate that the security administrators can enforce tenant-specific and fine-grained access control to protect cloud infrastructure assets, significantly improving multi-layer and multi-domain security protection, and achieving security by design.

4.3 Related work

According to the ETSI reference model [73] (the left part of Fig. 4.1), the NFV architecture is conceptually divided into three different layers: (1) NFV Infrastructure (NFVI) provides fundamental computing resources in which network service can be executed; (2) VNFs layer maintains a collection of network functions implemented in software; and (3) NFV MANO is responsible for managing the full lifecylce of infrastructure resources, VNFs, and network services.

As NFV MANO plays a significant role in NFV environment, many research efforts have been paid to develop and implement NFV orchestration framework. As a result, many NFV orchestration frameworks have become available, most of which are built according to the ETSI NFV MANO specification [74]. Some existing frameworks have been developed by the industrial sector like [48, 6], which are implemented as a non-model driven-based approach. Also, some are developed as open-source platforms, *e.g.*, OpenMANO [223], OpenBaton [168], ONAP [163]. In particular, OpenMANO aims to provide a practical implementation of management and orchestration based on ETSI standard. OpenBaton is focused on the basic orchestration of NFVO and VNFM, enabling VNF deployment on the top of multiple cloud infrastructures, such as OpenStack [170] and Docker [61]. Also, ONAP (Open Network Automation Platform) has been recently launched by combining two orchestration

projects (open source ECOMP [9] and Open-O [56]) together to build a software platform for delivering service automation and policy-driven orchestration. Specifically, ONAP architecture contains several software subsystems that are part of two major architectural frameworks: (1) design-time framework, which provides development environment for defining and describing deployable assets; and (2) execution-time framework, which executes the rules and policies defined within the design-time framework. It is worth noting that, although these aforementioned platforms have been implemented under open-source license and compliant with ETSI NFV MANO specification, the current version of their data models are not fully developed in a model-driven manner using TOSCA standard.

Our work is focused on the open-source NFV orchestration platforms, in which their data models are defined based on model-driven structure using TOSCA standard. For instance, Tacker [219], a TOSCA-based NFV orchestrator, has emerged as official OpenStack project to build NFV orchestration software supporting both NFVO and VNFM. In VIM module, Tacker uses Heat [104], a core part of OpenStack platforms, as a service orchestrator to deploy virtual resources over OpenStack infrastructure. It also maintains all the related NFV workflows and descriptors within the NFV catalog. These descriptors are defined according to TOSCA data model, and used by NFVO and VNFM for network service and VNF execution. Cloudify is another open-source TOSCA-based NFV orchestration framework [47], which aims at developing NFVO and VNFM to manage and orchestrate cloud infrastructure resources, VNFs, and network services. In Cloudify, the service templates and all the related configuration files are created based on TOSCA data model, called *Blueprints*, which are used for describing topology, components, relationships, and deployment plan, thereby making it possible to manage infrastructure as code.

After a careful study on the representative NFV orchestration frameworks, especially the open-source ones based on TOSCA data model (*e.g.*, Tacker, Cloudify), we observe that the given data models are mainly used to describe service orchestration plans, resource allocation, and basic lifecycle management of each network service. For example, it defines how the node is deployed, how the link relationships between the nodes are created, and how the workload is made up. The case study and comparative analysis of the NFV orchestrators indicate that,

1. The existing NFV orchestrators lack a dedicated module or component that can provide holistic security management, and support dynamic and centralized security control with high-level security policy specification. For example, Tacker itself does not have the capability of managing security mechanisms. Instead, it relies on the underlying OpenStack services such as Keystone [171] to manage its identity and authentication, and uses Newtron [157] to maintain its security group. Also, Cloudify creates secure communication by simply using TLS/SSL, allowing the clients to validate the authenticity of Cloudify Manager and ensuring that the data is sent in an encrypted mode.

2. Although the authors of [116, 7, 81] have already raised the security concerns in NFV orchestration and proposed security orchestrator for managing security mechanisms in the cloud-, IoT- and NFV-based architectures, the detailed data models and related use cases were not sufficiently conducted. Also, a software-defined security orchestration was proposed in [135], which aims at managing virtualized network security functions (*e.g.*, virtual firewall). However, their data models are not defined in a model-driven approach using TOSCA template.

3. To the best of our knowledge, TOSCA-based security model has not yet been formally defined in the existing NFV orchestration platforms which use TOSCA data models. Meanwhile, security attributes of interest cannot be explicitly defined for each virtual component (*e.g.*, VM, VNF), and there is no such a formal way to leverage the defined security attributes for achieving high-level security management.

Considering the potential widespread adoption of NFV in the clouds, the identified issues would eventually lead to the complete loss of the control over the deployed resources and NFV-based network services. It is therefore extremely significant and urgent to develop a dedicated security orchestrator to improve the existing NFV orchestrators with flexible security management, providing multi-layered security protection. In particular, we envision that such a security orchestrator can dynamically manage and orchestrate different security functions (*e.g.*, access control, data protection) in a holistic way, based on the particular needs of the tenants and users. In the following section, we will showcase how a tenant-specific access control with the proposed security orchestrator can be implemented in multi-cloud environment.

4.4 Design Motivation and Challenges

The previous discussion clearly indicate that the existing NFV orchestration frameworks mainly focus on maintaining the basic operations of service orchestration, as well as the lifecycle management of underlying infrastructure resources, VNFs, and network services, while the capability of providing holistic security management is limited. One interesting use case that motivates us to develop a dedicated security orchestrator is network slicing in 5G – a novel NFV-based approach to allowing network operators to provide dedicated virtual networks with functionality specific to the service or customer (called *network slice*) over a common shared network infrastructure. Examples of resources to be partitioned or shared would be bandwidth on a virtual network link, processing capacity of network elements (*e.g.*, VMs, VNFs) and storages. In such a scenario, multiple virtual networks or network slices are created and customized to meet the particular needs of different applications, services, devices, customers or operators. Within each slice, the resources are segmented and isolated into micro-segments (also called tenant domain) for realizing different network functions. To securely manage the network, each micro-segment needs to specify its own access control model and policies by taking into account its particular role. Unfortunately, due to the lack of holistic and flexible security management capability, the typical NFV orchestrators are not able to do this. With the objectives in mind, the following challenges need to be addressed in order to develop a dedicated security orchestrator.

- **Formally specifying security attributes of interest.** As aforementioned, NFV MANO plays a vital role in automatically deploying infrastructure resources (*e.g.*, VMs) and dynamically initiating VNF instances. This module usually uses service model templates like TOSCA to describe resource deployment and operational workflows for launching network service instances. However, the typical model templates including TOSCA neither specifies fine-grained security attributes for each virtual component (*e.g.*, VM, VNF) nor defines service templates with coarse-grained security policies. It is therefore necessary to formally extend the data models, ensuring that all the nodes presented in the template are associated with certain security attributes, thereby allowing high-level access control policies to be specified. It is worth noting, however, that such an extension should not incur any conflict or inconsistency with the original service template.

- **Multi-layer and multi-domain protection.** As shown in Fig. 4.1, NFV architectural framework contains three layers: NFVI, VNFs, and NFV MANO. The access to the assets at all these three layers must be well controlled. From north-south perspective, the implementation of one network service may rely on a pool of resources spanning from NFV infrastructure, platform, and VNFs. From west-east perspective, the resources used by a tenant can be distributed across different data centers and domains. Clearly, spotting the most appropriate granularities to enforce access control policies is a challenging is-

sue. Although this challenge also exists in the cloud environments, the extremely dynamic nature of NFV makes it more difficult.

- **Tenant-specific and on-demand specification of access control model and policies.** Cost-saving and flexibility make NFV a promising approach to implementing cloud-based systems, which are usually across multiple data centers. The NFV resources and network services (*e.g.*, VMs, VNFs) deployed for one tenant may be different from those of other tenants, resulting in the need that an access control model and the associated policies must be specific to the individual tenant. Unfortunately, the existing access control models, such as Mandatory Access Control (MAC) [120], Role-based Access Control (RBAC) [83, 204, 131], and Organization-based Access Control (ORBAC) [122], cannot meet such design requirements. By leveraging NFV orchestrator, we propose a software-defined access control paradigm which consists of an access control meta-model and policies, to achieve tenant-specific access control. More importantly, such an access control paradigm allows the customers to specify their own access control models and policies at the NFV service deployment stage via TOSCA templates.
- **Dynamic policy updates and fine-grained policy enforcement.** In the cloud, as one tenant can be created and removed on the fly, the corresponding access control model and policies need to be dynamically specified. That says, the access control policies that are initially generated from the extended TOSCA templates can be further dynamically updated or modified in the execution time. The challenge arising here is that, first, the security orchestrator must provide a friendly interface to re-specify and update access control policies. Second, the access control paradigm needs to support all the related operations, *e.g.*, update, modify, insert, delete. In addition, the access control policies need to be enforced at the appropriate PEPs (Policy Enforcement Points), *e.g.*, NFV infrastructure, VNFs. To do that, the access control paradigm needs to allow the fine-grained specification of policy rules. It is also necessary to have ”hook” built in the PEPs, so their behavior can be controlled by the policy rules, *e.g.*, which port of one VM can be opened, which VMs are allowed to communicate between each other.

4.5 Security Orchestrator

This section reports the design of our security orchestrator in detail. First of all, we describe the TOSCA data model and its extension for incorporating security attributes. We then present the design architecture and major components of our security orchestrator.

4.5.1 TOSCA model and its extension

In the cloud, data model (service template) plays a vital role in improving service automation during the process of installation, deployment, and management of cloud applications. Similarly, TOSCA data model plays such a role in NFV management and orchestration. Generally, a TOSCA file defines the building blocks of nodes and links (*e.g.*, types, properties, operations) that are used for constructing end-to-end network services. But the security attributes of each VM/VNF are either missing or not being presented in a desirable way, making it hard to specify access control policies. Therefore, we propose to extend the typical TOSCA data model with the security attributes of interest. In the current extension, two basic types of service template are mainly considered, while more service types can be incorporated in future extensions.

- VM description, which is used to deploy, configure, and manage the VM resources; and
- VNF description, which is used to initiate the VNF instances over the deployed VMs.

```
Example
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0
description: TOSCA-based security model for VM description
topology_template:
  node_templates:
    VM1:
      type: tosca.nodes.Compute
      capabilities:
          num_cpus: 2
          mem_size: 4096 MB
          disk_size: 2 GB
      properties:
          image: cirros-0.3.5-x86_64-disk
          availability_zone: nova
          vm_security_level: low
          tenant_domain: tenant1
          members: {(user1, high), (user2, medium), ..., (user5, low)}
    VM2:
      type: tosca.nodes.Compute
      capabilities:
          num_cpus: 2
          mem_size: 4096 MB
          disk_size: 2 GB
      properties:
          image: cirros-0.3.5-x86_64-disk
          availability_zone: nova
          vm_security_level: medium
          tenant_domain: tenant1
          members: {(user1, high), (user2, medium), ..., (user5, low)}
    VM3:
    VM4:
    ...
    VM10:
    #More VMs can be created
  security_group:
    VM_SP1:
      type: tosca.groups.nfv
      description: defining security group for tenant1
      target: {VM1, VM2, VM3, VM4, VM5}
    VM_SP2:
      type: tosca.groups.nfv
      description: defining security group for tenant2
      target: {VM6, VM7, VM8, VM9, VM10}
```

Figure 4.2: An example of extended TOSCA template for VM description (the extended security attributes are in bold)

Usually a TOSCA specification for VM covers the type (*e.g.*, compute service (Nova), storage (Swift), networking (Neutron)), capability (*e.g.*, sizes of CPU, disk, and memory), and properties (*e.g.*, image, flavor). A set of security related attributes are then defined, including security level, tenant domain, members, and security group, as shown in Fig. 4.2. They are used to generate access control policies via security orchestrator. For example, referring to the security level and a group of security policy defined in the VM description, the security orchestrator can generate the access control rules specifying whether the request to access to some particular resources are allowed or not. In the given example, VM_1 (which has 'low' security level) is not allowed to access VM_2 (which has 'medium' security level). Also, the attributes with respect to tenant domain allow the security orchestrator to identify in which tenant the VMs are deployed, so as to generate corresponding access control rules.

Another TOSCA example is about VNF, as shown in Fig. 4.3. Specifically, ten different types of VNFs are defined, and they are deployed and configured in different VMs. Also, each VNF is defined with a set of security attributes and a group of security policy with respect to tenant domain. In particular, five VNFs (from VNF_1 to VNF_5) are initiated in $tenant1$, while another five (from VNF_6 to VNF_{10}) are initiated in $tenant2$. As such, the security orchestrator will generate the access control rules that allow VNF_2 to access to the resources of VNF_1, because (1) the two VNFs are in the same tenant security domain, and (2) the security level of subject VNF_2 (medium) is higher than that of object VNF_1 (low).

```
Example
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0
description: TOSCA-based security model for VNF description
topology_template:
  node_templates:
    VNF1:
      type: tosca.nodes.nfv.VNF
      properties:
          type: server
          floating_ip: 10.0.0.122
          tenant_domain: tenant1
          vnf_security_level: low
          members: {(user1, high), (user2, medium), ..., (user5, low)}
      requirements:
          - virtualLink: subnet0
          - vm: VM1
      interfaces:
          lifecycle:
             create: install-server.sh
             start: start-server.sh
             stop: stop-server.sh
    VNF2:
      type: tosca.nodes.nfv.VNF
      properties:
          type: client
          floating_ip: 10.0.0.116
          tenant_domain: tenant1
          vnf_security_level: medium
          members: {(user1, high), (user2, medium), ..., (user5, low)}
      requirements:
          - virtualLink: subnet0
          - vm: VM2
      interfaces:
          create: install-client.sh
          configure: configure_client_to_server.sh
      relationships:
          type: tosca.nodes.relationships.ConnectsTo
          - target: VNF1
    VNF3:
    VNF4:
    ...
    VNF10:
    # more VNFs can be created
    subnet0:
      type: tosca.nodes.nfv.VL
      properties:
          ip_version: 4
          cidr: '10.0.0.0/24'
          gateway_ip: '10.0.0.1'
    subnet1:
      type: tosca.nodes.nfv.VL
      properties:
          ip_version: 4
          cidr: '10.0.1.0/24'
          gateway_ip: '10.0.1.1'
  tenant_security_group:
    VNF_SP1:
       type: tosca.groups.nfv
       description: defining security group for tenant1
       target: {VNF1, VNF2, VNF3, VNF4, VNF5}
    VNF_SP2:
       type: tosca.groups.nfv
       description: defining security group for tenant2
       target: {VNF6, VNF7, VNF8, VNF9, VNF10}
```

Figure 4.3: An example of extended TOSCA template for VNF description (the extended security attributes are in bold)

4.5.2 Design architecture and major components

As aforementioned, the purpose of our security orchestrator is to improve the capability of security management in NFV. Thus the security orchestrator is developed as an extension of the NFV orchestrator, and needs to comply with the ETSI NFV MANO architectural framework (as shown in Fig. 4.1). Specifically, its major operations include verifying and extracting security attributes from TOSCA file, generating access control policies by an access control engine, and finally enforcing the created policy rules at the appropriate enforcement points, *e.g.*, VMs, VNFs. The design architecture of our security orchestrator is illustrated in Fig. 4.4, and the major functional components are described in the following.

- *NFV orchestrator:* which maintains the lifecycle management of infrastructure resources and network services. It takes the standard TOSCA template as an input and constructs application topology by allocating virtual resources, creating the corresponding data cen-

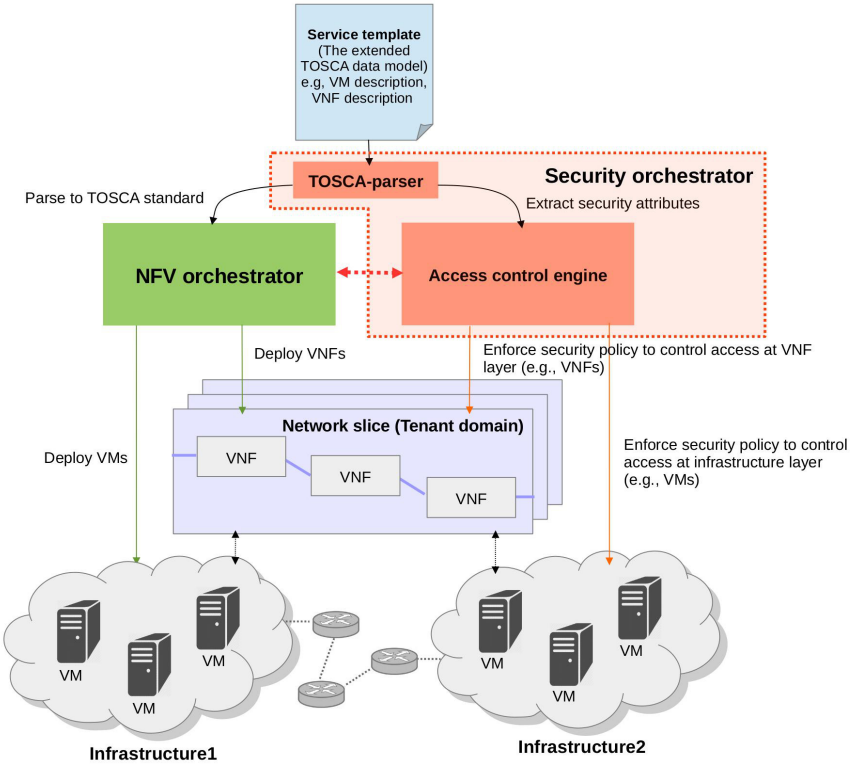


Figure 4.4: Design architecture of security orchestrator

ter and network connection, spinning up the VMs, loading the appropriate VNF software images, and finally connecting them together based on network forwarding graph for delivering end-to-end network services. As indicated in Fig. 4.4, the NFV orchestrator interacts with Virtualized Infrastructure Manager (VIM) to allocate VMs, and interacts with VNF Manager (VNFM) to instantiate the VNFs[1].

- *Security orchestrator:* which works together with the NFV orchestrator to provide monitoring, resource access control, and security policy enforcement to the deployed components, *e.g.*, VMs, VNFs. Specifically, it extracts the security attributes of each virtual component from the extended TOSCA data model and creates the corresponding security policies based on the tenant-specific access control model, and further enforces these policies to control resource access at infrastructure and VNF layers, *e.g.*, VMs, VNFs. In particular, the security orchestrator is composed of two modules:

  1. *TOSCA-parser:* which transforms high-level configuration (*e.g.*, abstract flow, service definition) that operators architect the networks into specific configuration parameters for initiating network services. Specifically, it firstly extracts the security attributes of VM/VNF nodes from a given TOSCA file, then parses them to the access control engine in order to generate access control policies based on the tenant-specific access control model. Such a translation example is given in Section 4.5.1. Meanwhile, the necessary information other than security attributes are input to the NFV orchestrator.
  2. *Access control engine:* which allows for specifying tenant-specific access control model, and generating the corresponding access control policies by taking into account the security attributes extracted from the TOSCA file. More details about the access control engine is given in section 4.6.

[1]For simplicity in explanation, VIM and VNFM have been implicitly hidden within the infrastructure and VNF layers, respectively.

4.6 Access Control Engine

This section is devoted to presenting the central part of our security orchestrator, *i.e.*, access control engine. A design framework is firstly given, followed by the design principles.

4.6.1 Access control framework

Our security orchestrator employs an access control engine [173, 103] to specify high-level security policies, which are used to control the access to the resources of particular cloud tenants. As illustrated in Fig. 4.5, the TOSCA-parser extracts security attributes of each VM/VNF from the extended TOSCA file, and parses these parameters to access control engine for further generating access control rules. Specifically, the predefined security attributes of each VM/VNF are assigned to *subject* and *object* according to attribute-based policy specification. Then the requests are granted to access the resources if the attributes of *subject* and *object* match with the rules defined in the access control policy. In particular, the access control framework can be classified into three levels which are described as follows.

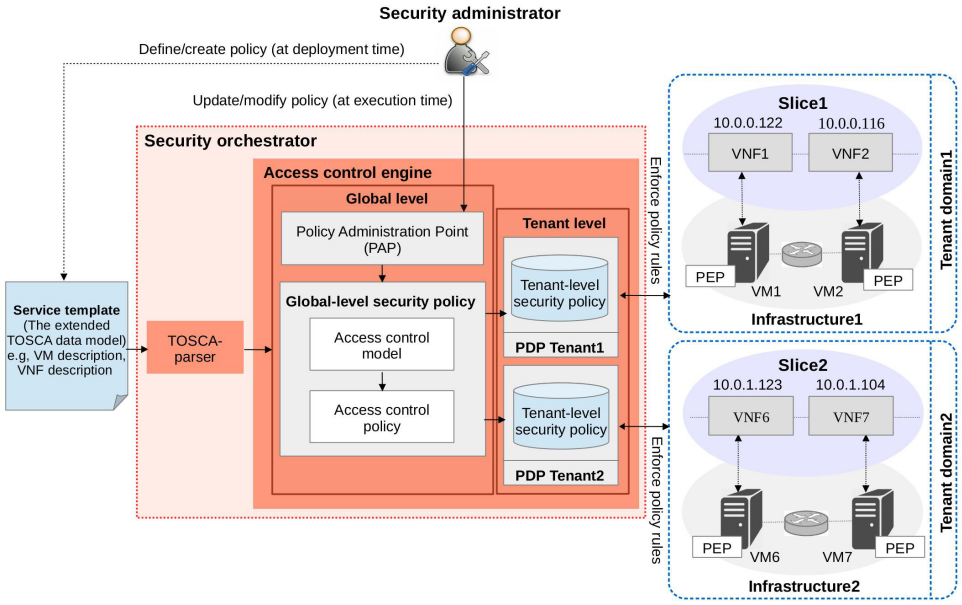


Figure 4.5: Design framework and operational workflow of access control engine

- *Global level:* which maintains two main components, *Policy Administration Point (PAP)* and *global-level security policy*. The PAP provides security policy specification interface, allowing security administrators to create, update, modify or reconfigure the global security policies or the tenant-specific security policies at execution time. For example, when security requirements are changed with respect to the security attributes of VM/VNF. Meanwhile, the global-level security policy contains multiple access control models and policies, each one is dedicated to one tenant domain. It holds a global view of all the tenant information related to authorization policies and rules. Thus, the changes operated on the global security policy must be distributed to all tenant-level security policies.

- *Tenant level:* there is a *tenant-level security policy* which contains a specific set of access control policies and all the related information about the local resources usage of one tenant. Each tenant has it own *Policy Decision Point (PDP)* which is considered as the main decision point for access requests. Once the PDP receives the requests from

its local *PEPs (Policy Enforcement Points)*, it validates these requests based on information gathering from the tenant-level security policy, and finally conclude the authorization decisions.

- *Enforcement level:* each instantiated resource in the cloud infrastructure (*e.g.*, VM, VNF), embeds a dedicated PEP. The PEP is in charge of sending authorization requests to the corresponding PDP and protecting the tenant's resources by enforcing appropriated access control rules based on the authorization decisions.

4.6.2 Tenant-specific access control paradigm

The access control engine provides a tenant-specific access control paradigm, which allows the security administrators to dynamically customize the access control models (*e.g.*, DAC, MAC, RBAC [120]) and policies based on the specific tenant requirements in the cloud. To present such a paradigm, two notations are given as follows.

- *Entity:* which refers to the users or cloud resources with respect to the cloud tenant, and they can be represented as *subject* or *object* in the access control model. In any access control model, *subjects* refer to the entities that can perform actions on the system, and *objects* refer to the resources and network services to which access need to be controlled. In the cloud environment, *subjects* can be automatically generated (*e.g.*, VM and VNF instances), which are possibly chained together for creating end-to-end network services.
- *Attribute:* which is considered as meta-data, representing entity specification such as properties, capabilities, and relationships. In particular, one entity can be assigned with several types of attributes. Based on our contribution, the attributes here are those security-related properties, such as security level, tenant domain, a group of security policy, and the type of resources/services. We can denote $Attribute = Category \times CatValue$; where $Category$ is a set of security-related properties (*e.g.*, security-level), and $CatValue$ is a set of potential values of each related category (*e.g.*, security-level $\big[$low, medium, high$\big]$). These attributes are eventually used to create the corresponding access control policy rules.

Our motivation to develop a tenant-specific access control paradigm is that, although many access control models are currently available (*e.g.*, DAC, MAC, RBAC [120]), they are fundamentally lack of flexibility on specifying different models and policies. For example, RBAC can be applied to one tenant, but it cannot be used to multiple tenants which may be varied in access control requirements. Therefore, we propose a software-defined access control paradigm, which consists of an access control model (**ACM**) and its corresponding access control policies (**ACP**), as illustrated in Table 4.1. Specifically, the ACM contains meta-data (MD) and meta-rule (MR), to enable different tenants to arbitrary specify their own access control models and policies based on the specific needs. The MD defines a schema to instantiate an access control model and the MR defines a schema to create the rules. Then the corresponding ACP is created based on a tenant-specific access control model. In particular, the ACP specifies a set of entities, *i.e.*, *subject*, *object*, and *action*. For simplicity of use here we called it as data (D). It then assigns the entity-data assignment values to each entity (EDAss), which can further obtain from category value $CatValue$. According to Table 4.1, $SubjectDataAss, ObjectDataAss, ActionDataAss$ are respectively represented the entity-data assignment values of each *subject*, *object*, and *action*. Next, it identifies the perimeter (P) in which a set of entities needs to be protected, and finally establishes the rules (R). Formally, it takes *subject*, *object*, and *action* to evaluate an authorization decision (grant or deny). Thus, the requests are granted to access the resources if attributes of *subject* and *object* match with the created rules.

Table 4.1: Basic sets and functions of access control model and policy

Definition:
Meta-data (MD): defines a schema to instantiate an access control model.
Meta-rule (MR): defines a schema to create the rules.
Data (D): contains a set of entities, *i.e.*, subject, object, and action.
Entity-data assignment (EDAss): assigns entity-data assignment values to each entity
Perimeter (P): specifies a set of entities to be protected
Rule (R): establishes access control rules based on attributes, *i.e.*, security-related properties of subject, object, and action
Category: maintains a set of security-related properties, *e.g.*, security-level
CatValue: defines a set of values of each category, *e.g.*, security-level [low, medium, high]

(1) Access control model (ACM) = (MD, MR)
- $MD : \{SubjectMD, ObjectMD, ActionMD\}$
  where; $SubjectMD, OjectMD, ActionMD \subseteq Category$

- $MR : \{SubjectCategory \times ObjectCategory \times ActionCategory\} \rightarrow Instruction$
  where; $SubjectCategory \subseteq SubjectMD$
  $ObjectCategory \subseteq ObjectMD$
  $ActionCategory \subseteq ActionMD$
  $Instruction : \{AuthzDecision\}$

(2) Access control policy (ACP) = (D, EDAss, P, R)
- $D : \{SubjectD, ObjectD, ActionD\}$
  where; $SubjectD \subseteq \{SubjectMD\}$
  $ObjectD \subseteq \{ObjectMD\}$
  $ActionD \subseteq \{ActionMD\}$
- $EDAss : \{SubjectDataAss, ObjectDataAss, ActionDataAss\}$
  where; $SubjectDataAss \subseteq \{SubjectD \times CatValue\}$
  $ObjectDataAss \subseteq \{ObjectD \times CatValue\}$
  $ActionDataAss \subseteq \{ActionD \times CatValue\}$
- $P : \{S, O, A\}$
  where; $S \subseteq \{SubjectD\}$
  $O \subseteq \{ObjectD\}$
  $A \subseteq \{ActionD\}$
- $R : \{S \times SubjectDataAss, O \times ObjectDataAss, A \times ActionDataAss\} \rightarrow \{Instruction, [grant, deny]\}$

4.6.3 Use case

In this section, we develop a use case to illustrate the use of an extended TOSCA file to generate tenant-specific access control policies using our proposed access control paradigm.

TOSCA-based policy specification. Referring to the TOSCA files shown in Fig. 4.2 and Fig. 4.3, ten users and ten different types of VMs and VNFs, as well as two groups of security policy are specified. The defined security attributes are then extracted and populated into the parameters of our proposed access control paradigm, as shown in Table 4.2. In particular, perimeter P is used to identify the scope of protection, isolating a set of entities from one tenant to other tenants. For example, perimeter P_1 contains the entities lying in $tenant1$, *e.g.*, $\{VM_1, VM_2, ..., VM_5\}$, $\{VNF_1, VNF_2, ..., VNF_5\}$, and $\{user_1, user_2, ..., user_5\}$. Similarly, perimeter P_2 contains the entities of $tenant2$, *e.g.*, $\{VM_6, VM_7, ..., VM_{10}\}$, $\{VNF_6, VNF_7, ..., VNF_{10}\}$, and $\{user_6, user_7, ..., user_{10}\}$. A set of access control rules (R_t) are then created based on these perimeters, security-related properties (*i.e.*, security-level of subject and object), and the entity-data assignment values (*i.e.*, [high, medium, low]). Regarding to the given example, $t = \{1, 2\}$ denoting as $tenant1$ and $tenant2$ respectively. As shown in Table 4.2, each rule set contains multiple rules which can be represented in the form of $R_t = \{r_1, r_2, r_3, ..., r_n\}$; where

Table 4.2: Use case: generating tenant-specific access control policies

Data (D):

$SubjectD$ = {(user$_1$, user$_2$, ..., user$_{10}$), (VM$_1$, VM$_2$, ..., VM$_{10}$), (VNF$_1$, VNF$_2$, ..., VNF$_{10}$) }

$ObjectD$ = {(VM$_1$, VM$_2$, ..., VM$_{10}$), (VNF$_1$, VNF$_2$, ..., VNF$_{10}$)}

$ActionD$ = {action-type, (vm-action, vnf-action)}

Entity-data assignment ($EDAss$):

$SubjectDataAss$ = {($user_1$, high), ($user_2$, medium), ..., ($user_{10}$, low), (VM$_1$, low), (VM$_2$, medium), ..., (VM$_{10}$, high), (VNF$_1$, low), (VNF$_2$, medium), ..., (VNF$_{10}$, high)}

$ObjectDataAss$ = {(VM$_1$, low), (VM$_2$, medium), ..., (VM$_{10}$, high), (VNF$_1$, low), (VNF$_2$, medium), ..., (VNF$_{10}$, high)}

$ActionDataAss$ = {(vm-action, access-vm), (vnf-action, access-vnf)}

Perimeter (P_1): for $tenant1$

S_1 = {user$_1$, user$_2$, ..., user$_5$, VM$_1$, VM$_2$, ..., VM$_5$, VNF$_1$, VNF$_2$, ..., VNF$_5$}

O_1 = {VM$_1$, VM$_2$, ..., VM$_5$, VNF$_1$, VNF$_2$, ..., VNF$_5$}

A_1 = {(access-vm), (access-vnf)}

Rule set (R_1): for $tenant1$

r_1 = {($user_i$, high), (VM$_j$, [high, medium, low]), (access-vm)} → {(instruction, grant)}

r_2 = {($user_i$, medium), (VM$_j$, [medium, low]), (access-vm)} → {(instruction, grant)}

r_3 = {($user_i$, medium), (VNF$_j$, [medium, low]), (access-vnf)} → {(instruction, grant)}

r_4 = {($user_i$, low), (VNF$_j$, low), (access-vnf)} → {(instruction, grant)}

r_5 = {(VM$_i$, high), (VM$_j$, [high, medium, low]), (access-vm)} → {(instruction, grant)}

r_6 = {(VM$_i$, medium), (VM$_j$, [medium, low]), (access-vm)} → {(instruction, grant)}

r_7 = {(VNF$_i$, medium), (VNF$_j$, [medium, low]), (access-vnf)} → {(instruction, grant)}

r_8 = {(VNF$_i$, low), (VNF$_j$, low), (access-vnf)} → {(instruction, grant)}

n indicates a number of rules. For all $r_n \subseteq R_t$, the rules are created as;

$$\{S_t \times SubjectDataAss, O_t \times ObjectDataAss, A_t \times ActionDataAss\} \rightarrow \{(instruction, grant)\}.$$

The subject can be either users, VMs or VNF instances, and the object refers to the VMs and VNFs to which access need to be controlled. For example, r_1 = {($user_i$, high), (VM$_j$, [high, medium, low]), (access-vm)} → {(instruction, grant)}. Thus the request is granted if the security-level of subject (*e.g.*, user$_i$) is equivalent or higher than the security-level of object (*e.g.*, VM$_j$), otherwise the access is denied. As shown in table 4.2, thanks to tenant-specific access control paradigm, multi-layer and multi-domain protection can be achieved.

This use case clearly indicates that the requests to access the resources are granted if and only if one of the created policy rules is matched. In another word, only legitimate requests in the same perimeter (or the same tenant domain), where their entity-data assignment values match with some given rules, are granted. Otherwise, the access will be denied.

Dynamic policy update. In addition to generating access control policies based on the security attributes defined in the TOSCA data model, the tenants and security administrators should be able to dynamically configure the access control models and update the access control policies. Relying on the static configuration of security attributes in the extended TOSCA files at the deployment phase may not achieve the objectives, as frequently modifying the TOSCA files and load them into the orchestrator is not an effective solution. The access control engine is therefore expected to integrate the advanced control features to allow the administrators to dynamically update or reconfigure the security policies during their execution times.

In Fig. 4.5, two different access control models are specified, respectively containing a set of access control policy rules to be enforced in the corresponding tenant domain. During the management lifecycle of VMs and VNFs, a security administrator can update the policy rules

through the PAP. For example, if the security administrators intend to change security requirements thereby: (1) adding a new entry in the global-level security policy to allow incoming SSH connections (port 22) regardless of the tenant domains; and (2) modifying the existing rules in the tenant-level security policy to allow specific incoming connections such as HTTP (port 80), while rejecting the request that does not match the condition even though it is generated within the same tenant domain (as shown in Table 4.3).

Table 4.3: An example of change requirement of security policy rule sets

Type	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
Global-level security policy	SSH	Any	*	Any	22	accept
Tenant-level security policy (Tenant1)	TCP	10.0.0.*	*	10.0.0.*	80	accept
Tenant-level security policy (Tenant2)	TCP	10.0.1.*	*	10.0.1.*	80	accept

Based on tenant-specific access control paradigm, these defined policies are assigned to *subject*, *object*, and *action*, in order to create the access control policy rules. For example, the rule w.r.t SSH connection is created as;

$$r_i = \{SubjectD, (ObjectD, (port, 22)), ActionD\} \rightarrow \{(instruction, grant)\};$$

in which *SubjectD* and *ObjectD* covers all the subjects and objects in the two tenant domains. Such a rule will be enforced to all the tenants, so each tenant needs to update the new rule in its tenant-level security policy. Similarly, the access control rules regarding to HTTP connection are defined in the following way,

$$r_j = \{S, (O, (port, 80)), A\} \rightarrow \{(instruction, grant)\};$$

which means that only HTTP connection requests coming from the same tenant are granted. This rule is enforced as tenant-level security policies for both $tenant1$ and $tenant2$.

4.7 Proof of Concept Validation

This section presents the implementation details of a prototype of our proposed security orchestrator, demonstrating its operation, together with NFV orchestrator to extract security attributes from TOSCA file, generate tenant-specific access control policies, and further enforce them to the cloud infrastructure.

4.7.1 Prototype development and implementation

To develop our security orchestrator prototype, we choose Tacker [219], an official Openstack project, as NFV orchestrator to manage infrastructure resources and maintain the lifecycle management of network services and VNF instances over the OpenStack infrastructure. Another important reason for using Tacker is that it supports TOSCA files. For security orchestrator, we integrate access control engine [173] to provide centralized security management, *e.g.*, create, update, and reconfigure tenant-specific security policies. Specifically, our security orchestrator contains three functional modules briefly described below. The hardware specification is given in Table 4.4.

- *Client-interface:* which is used to interact with access control engine via CLI. The communication with *Client-interface* is done via Python subprocess module.
- *TOSCA-parser:* which is designed for two purposes. First, it extracts the description of VM/VNF nodes, translates the security attributes defined in the extended TOSCA file, and

Table 4.4: Hardware specification

Role	Number of deployment	Hardware specification
NFV orchestrator	1	Intel(R) Core(TM) i5, 2.40 GHz CPU, 8 GB RAM
Security orchestrator	1	Intel(R) Core(TM) i7, 2.50 GHz CPU, 16 GB RAM
OpenStack server for running 10 VMs and initiating 10 VNFs	1	Intel(R) Core(TM) i5, 2.40 GHz CPU, 8 GB RAM

parses these parameters to access control engine for generating security policies. Second, it fills all information about the node-template part defined in the extended TOSCA file, translates them into standard TOSCA form and input to the existing NFV orchestrator.

- *Network hook:* which works as Policy Enforcement Point (PEP). It is installed in each VM/VNF, and sends authorization request to the access control engine. It is written in Python and maintained the connections between *subject* and *object*. By default, it disables all the accesses to the *object* by setting the rule in its iptables as `iptables -A INPUT -j DROP`. It continues listening to the assigned port, and checks the permission based on the source IP with access control engine when it receives the connection 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 -s 10.0.0.116 -j Accept`, so that the *subject* whose IP address is `10.0.0.116` starts communication with the requested *object*.

4.7.2 Feasibility studies

The operational flow of our security orchestrator is illustrated in Fig. 4.6. The current prototype implements a software-defined tenant-specific access control paradigm that was presented in section 4.6. The figure shows that, once an access request is received at the *object*, the corresponding network hook sends an authorization request to the access control engine for validation. If the request is authorized, the network hook will add a new rule into its `iptables` to allow this connection. During a predefined period of time, if the network hook does not receive any request then the corresponding rule will be removed from its `iptables`, and the connection between *subject* and *object* will be disconnected.

The testing result from implementation is presented in Fig. 4.7. As shown in the two telnet consoles, VNF_2 successfully establishes network connection with VNF_1, as they are in the same tenant domain, while the security level of VNF_2 (medium) is higher than the security level of VNF_1 (low). In another word, the cross group request, *e.g.*, the connection request from VNF_6 to VNF_1, is totally denied. This well illustrates that the network connection can be established only if the *subject* and *object* are matched with the access control rules defined in the security policy.

4.8 Performance Evaluation

In this section, we reports the experiments on running our security orchestrator on a practical testbed, with a purpose to evaluate its performance in terms of three primary metrics, i.e., throughput, scalability, and adaptability.

4.8.1 Experiment settings

As the core component of our current security orchestrator is tenant-specific access control paradigm, we particularly carried out a set of experiments to evaluate the performance of the

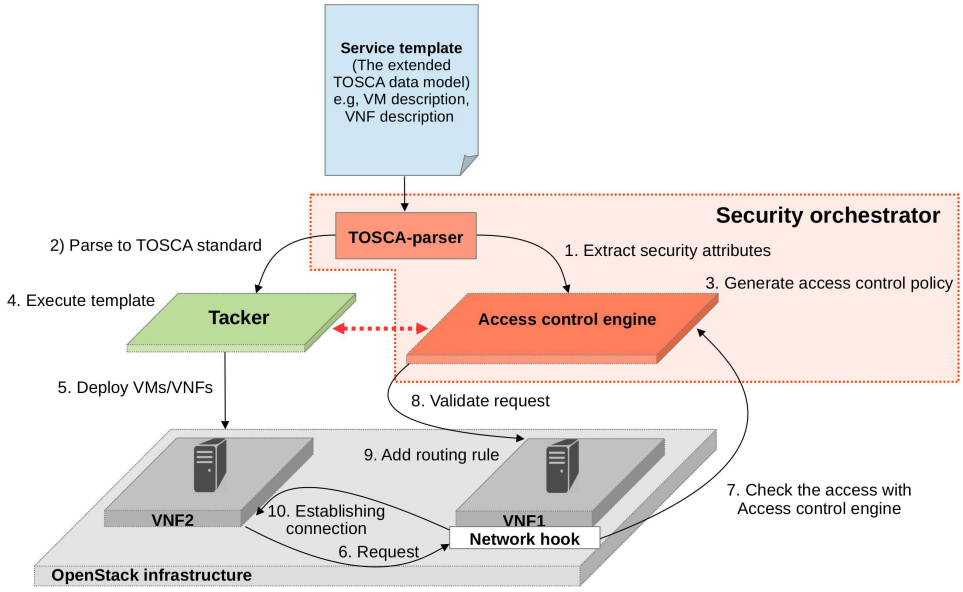


Figure 4.6: The operational flow of security orchestrator

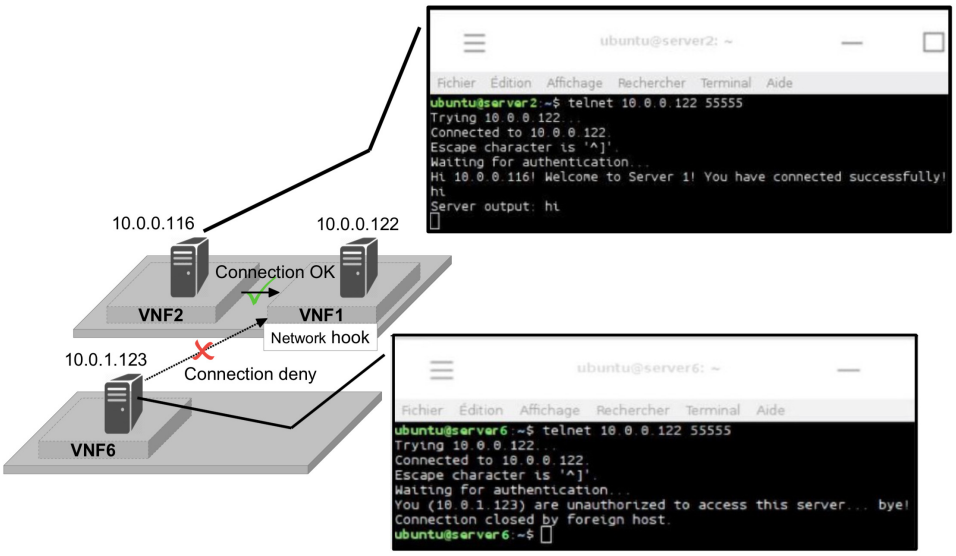


Figure 4.7: The result of testing network connection with Telnet

access control engine. In particular, the experimental testbed set up for the experiment contains three HA (High-Availability) OpenStack clusters, among which one serves as a master platform and two run as slave platforms. As shown in Fig 4.8, each slave platform is equipped with 5 VMs. The hardware configuration is listed in Table 4.5, where one master platform and two slave platforms were constructed using Dell R730 servers.

4.8.2 Results and analysis

As shown in Table 4.1, we used the meta-model to initiate different access control models (ACM) and policies (ACP). For brevity, *SMPolicy* is used here to compactly represent the access control model and its policies. We carried out the experiments on the established testbed and averaged the experimental results from multiple trials or iterations. In the following, we report our observations and findings in terms of three performance metrics that are essential to the access control engine of our security orchestrator.

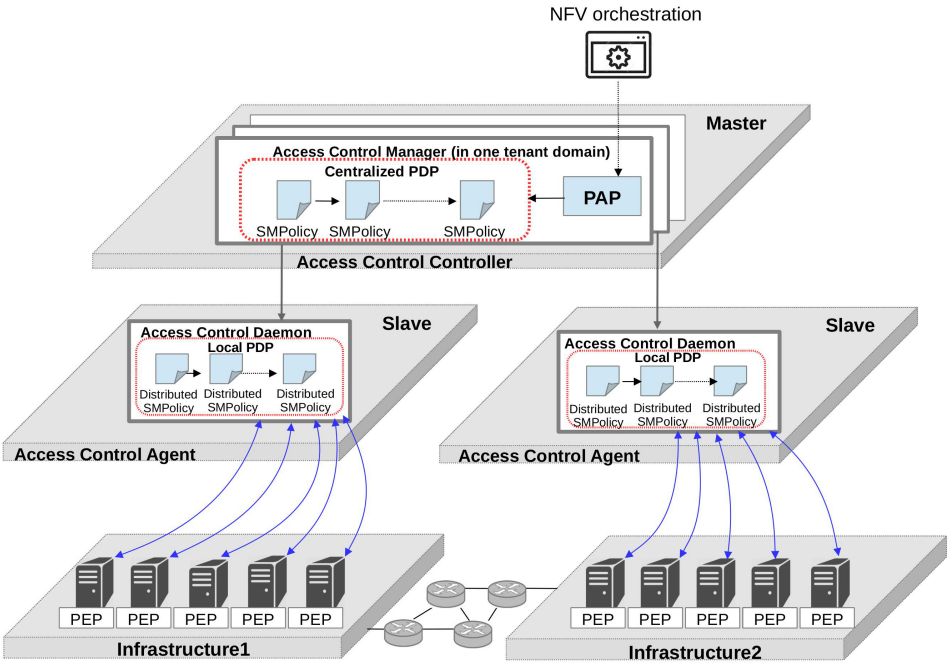


Figure 4.8: The deployment of access control engine: one master and two slave nodes

Table 4.5: Hardware specifications for master and slave platforms

Node	Number of deployment	Hardware specification
Master platform	1	Intel E5-2680 (48 cores/ 256GB RAM)
Slave platform	2	Intel E5-2630 (36 cores/ 128GB RAM)

4.8.2.1 Throughput

One of the key metrics for evaluating the capability of access control engine is throughput (transaction rate), namely the number of authorization requests per second that it can handle.

In the first experiment, we specified the *SMPolicy* as a basic Role Based Access Control (RBAC) model, which has 10 users, 5 roles, and 10 objects. All those security attributes were firstly defined in the extended TOSCA file at deployment time. We then gradually increased the number of requests to observe the throughput of the access control engine. As shown in Fig. 4.9, the throughput reached to the average rate 4.1 requests per second when the request frequency was increased from 1 to 20 requests per second. It is worth noting that, thanks to the micro-service architecture developed in the access control framework, one identical *SMPolicy* container can be automatically launched when the number of request is beyond the throughput of access control engine.

One of our hypotheses is that the throughput of access control engine could be potentially impacted by RAM and CPU. Therefore, in the second experiment, we varied the RAM capacity of the *SMPolicy* container and observed how the throughput of the access control engine could be impacted. Same as the previous experiment, we configured one tenant, and set *SMPolicy* as the basic RBAC with 10 users, 5 roles, and 10 objects, running with one CPU core. The result is depicted in Fig. 4.10a, which clearly indicates that the throughput remains stable at 2 requests per second even though if the size of RAM is increased.

Similarly, we evaluated the impact of CPU on the throughput by changing the number of cores. To conduct this experiment, we fixed the memory size as 4GB and used the same tenant configuration. We continually increased the number of CPU cores used by the *SMPolicy* container, and observed that the throughput always remained at 3 requests per second even when

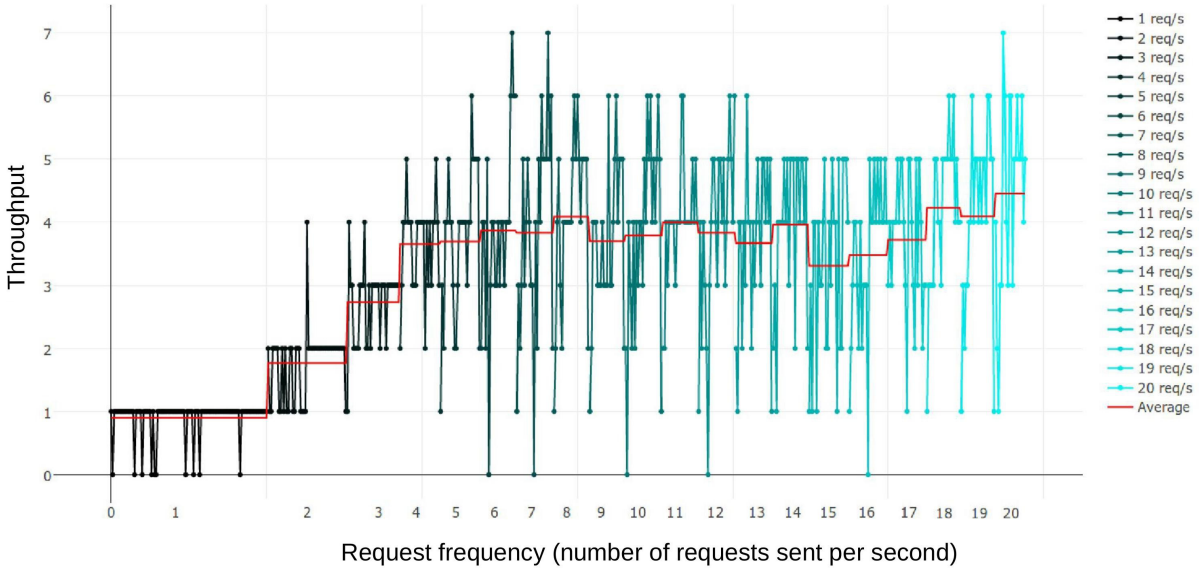


Figure 4.9: Throughput on handling the number of authorization requests

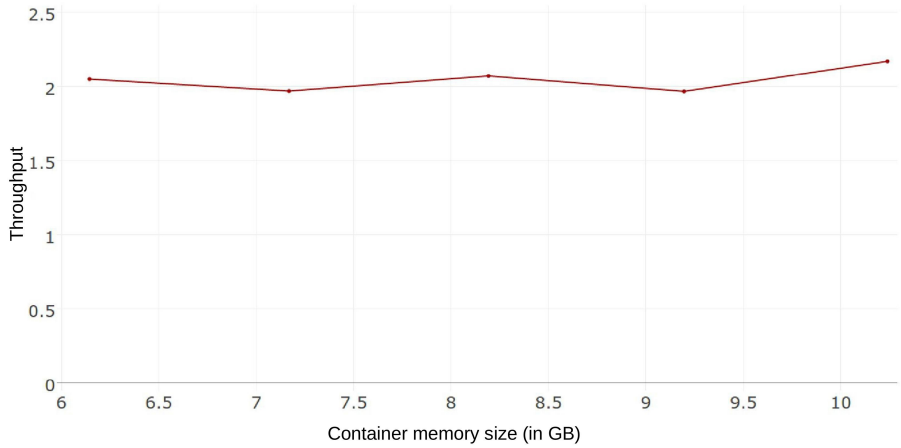
the number of CPU cores exceeds 2. The results are shown in Fig. 4.10b. The experiments clearly demonstrated that RAM and CPU have slight impacts on the throughput of access control engine.

4.8.2.2 Scalability

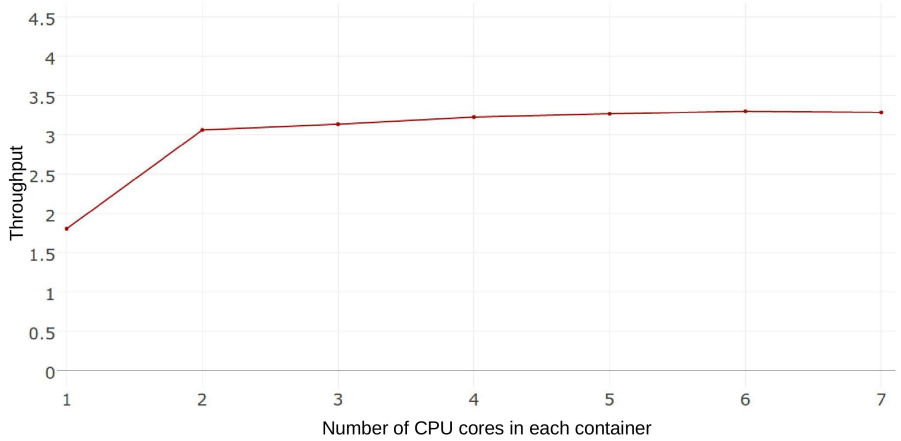
Another important concern about our security orchestrator is that whether or not it can maintain the satisfactory throughput with the increasing number of users and tenants. This can be viewed as scalability, which has been tested with two experiments.

Scalability with the increasing number of users. As the previous experiments, we initially set *SMPolicy* with the same configuration for one tenant (RBAC, 10 users, 5 roles, and 10 objects). We then increased the number of users and observed the throughput (transaction rate). As shown in Fig. 4.11a, the throughput remains stable as 5.9 requests per second when the number of users is less than 50. It then dramatically decreased when the number of users exceeds 50. The worst case is 0.5 requests per second when the number of users reaches to 1500. The main cause of sharp decreasing is that we developed the access control framework as the micro-service architecture which implemented through a set of containers. That's means, a single *SMPolicy* container is initially instantiated for handling the certain number of user requests. One identical *SMPolicy* container can be automatically launched when the number of requests go beyond the throughput of access control engine. However, our experimental result (as shown in Fig. 4.11a) is specifically evaluated based on a single *SMPolicy* container.

Scalability with the increasing number of tenants. As aforementioned, one of the salient features of our novel tenant-specific access control paradigm is that it allows to create and customize the access control policies for the newly generated tenants on the fly. To evaluate the throughput of access control engine with the varying number of tenants, we configured each tenant with the same *SMPolicy*. Fig. 4.11b shows that the throughput of access control engine varies from 5.7 requests per second (one tenant) to 4.5 requests per second (10 tenants), demonstrating slight degradation. The main reason is that our access control engine is implemented using the micro-service architecture, in which *SMPolicy* of each tenant run in the dedicated and independent containers, enabling it to scale freely to support multi-tenancy environment.



(a) Varying in size of RAM



(b) Varying in number of CPU cores

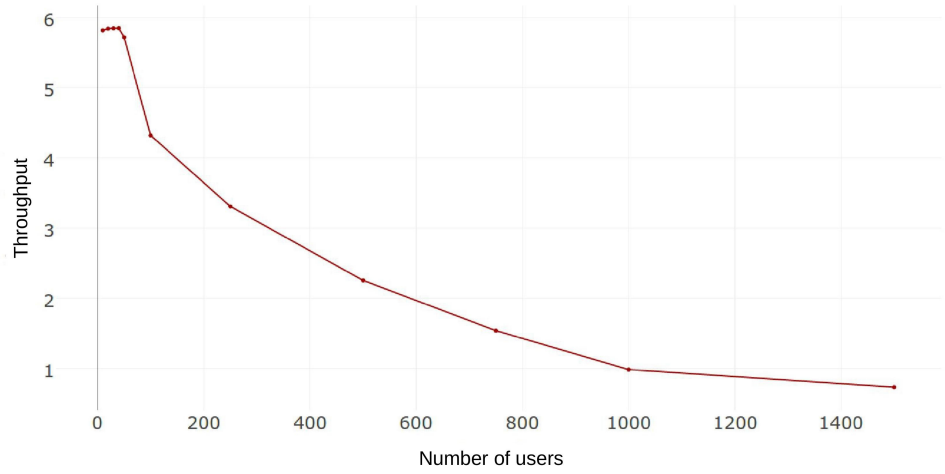
Figure 4.10: Average throughput with varying number of RAM and CPU cores

4.8.2.3 Adaptability

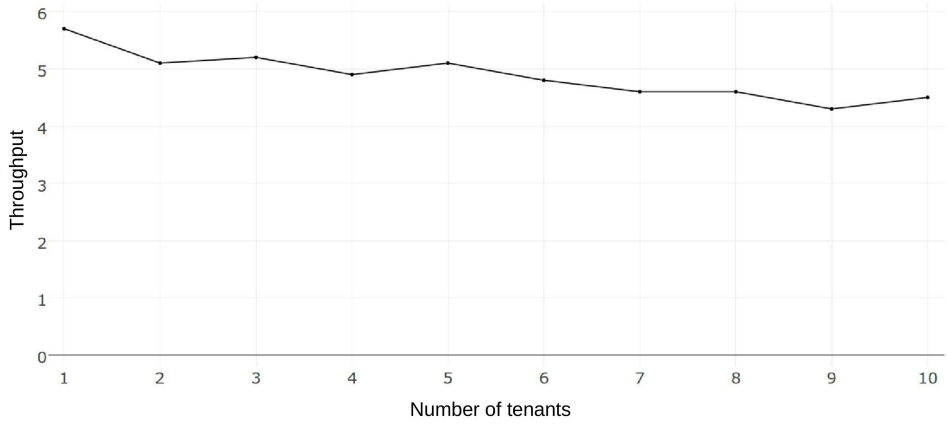
Our proposed access control meta-model decomposes a security policy into a set of components, which can be updated and synchronized independently instead of being handled as a whole. Thanks to this property, any update of global-level security policy (via a centralized PDP) can be effectively and timely achieved in the tenant-level security policy (via local PDPs) residing in different access control agents running in the distributed data centers. To evaluate this property, we set up the experimental environment with one server acting as master node and two servers running as slave nodes (as shown in Fig. 4.8), we then ran the experiments as follows.

We firstly set up two *SMPolicy* containers, which reside in the master node and one slave node respectively. We then modified the *SMPolicy* on the master node, and measured the duration for the slave node to completely update its local *SMPolicy* and return to the maximum throughput. The results are presented in Fig. 4.12, which shows that it took around 17 seconds on average for the slave node to return to its maximum throughput.

We are also curious how many slave nodes at most that one master node can handle. To answer this question, we had to measure the *average latency* that all the slave nodes can successfully update their local *SMPolicy* from the master node. Specifically, we still configured a tenant with the same *SMPolicy* which has 10 users, 5 roles, and 10 objects. We then gradually increased the number of slaves. As shown in Fig. 4.13, the master node could support 10 slaves



(a) Varying in number of users (from 10 to 1500)



(b) Varying in number of tenants (from 1 to 10)

Figure 4.11: Average throughput with the varying number of users and tenants

with very slight performance degradation, *i.e.*, the average adaptation latency varies from 17 seconds (one slave) to 19.8 seconds (10 slaves). Note that these numbers of slave are good enough to be considered when setting up a practical testbed. Because one slave node contains a local PDP which provides the access control decision regarding to one data center, so that 10 slaves can be equivalent of managing the access control over 10 distributed data centers.

4.9 Conclusion and Future Work

Although it is clear that NFV MANO can enable network resources and services to be managed and provided in agile, dynamic, scalable and adaptive ways, it remains unclear how it benefits security management, ensuring that all the desirable security properties of network services can be preserved in their entire lifecycles. In Chapter 2, we have proposed a conceptual framework SecMANO, but there are still many challenges to be tackled to have it implemented in real world. This Chapter is therefore devoted to implementing the idea exploring the feasibility and effectiveness of leveraging NFV orchestrator to develop a security orchestrator – as an actual implementation of SecMANO.

Our case studies on the existing NFV orchestration frameworks already showed that none of them provide dedicated modules or components for security management purpose, but they are open to be extended thanks to the structured data model like TOSCA (which has potential

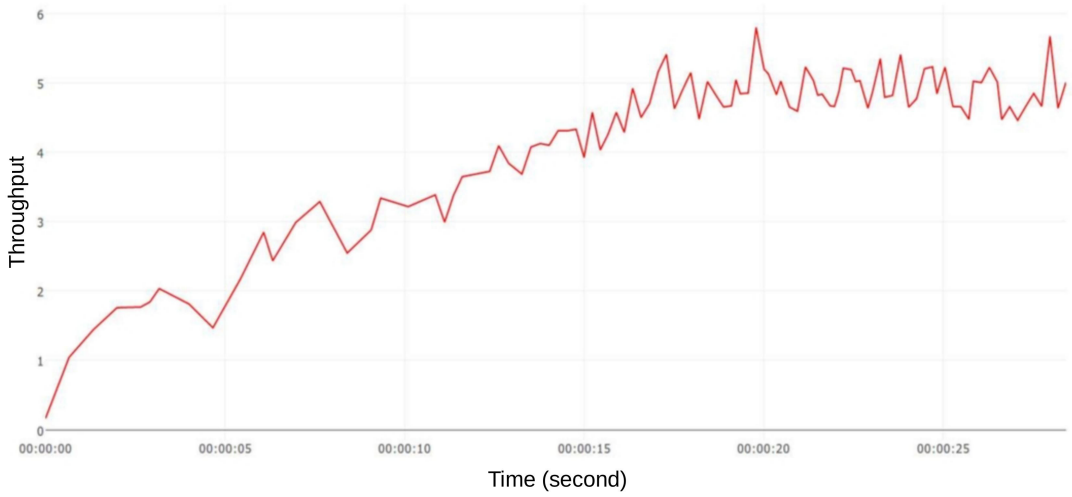


Figure 4.12: Time taken for the slave nodes to update *SMPolicy* from the master node

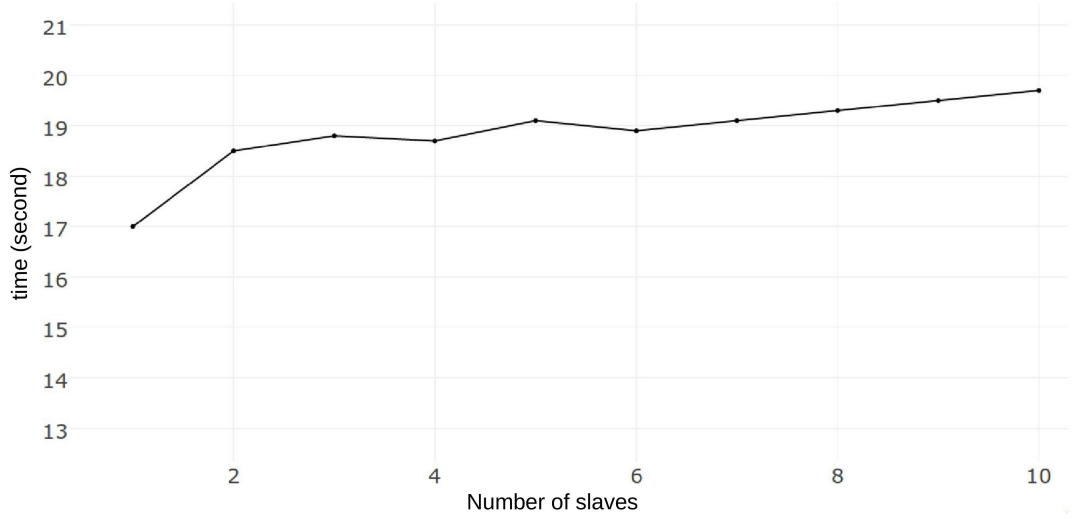


Figure 4.13: Adaptation period when the number of slave nodes is increased

to be widely used by today's NFV orchestrators). Thus, we extended the TOSCA data model to define security attributes of interest, which were then parsed to our security orchestrator to generate the corresponding security policies. The core component of our security orchestrator is a novel access control paradigm, which mainly contains an access control meta model that can be used to specify different access control models like RBAC and their policies, according to the needs of particular tenants.

For validation and evaluation purpose, we developed a prototype to demonstrate that the network connections between the VMs/VNFs residing in different security domains can be dynamically controlled. We then emulated three data centers and deployed our security orchestrator in this testbed to test its performance in terms of throughput, scalability, and adaptability. The experimental results indicated that creating on-demand, tenant-specific, and dynamic access control policy in a fine-grained manner in the clouds became possible, thanks to our software-defined access control engine. Interestingly, the throughput of our security orchestrator has always maintained at a satisfactory level, despite the increasing number of tenants, users, or objects that are deployed in the cloud.

However, the current version of security orchestrator only a particular access control model RBAC was evaluated. In the future work, we will further study more access control models

and their coexistence. We will also extend the capability of our security orchestrator to support other security functions, such as network isolation, data protection, IDS/IPS, to achieve a holistic security management orchestration. Meanwhile, as the community of NFV and SDN are very dynamic and keep evolving, we will keep eyes on the development of NFV MAMO framework, so as to leverage and integrate the key functional blocks, such as monitoring, policy engine, into our security orchestrator. From a long-run perspective, our security orchestration is expected to accommodate various security functions simultaneously, dynamically and optimally configuring and deploying security services on demand, in accordance with the high-level security policies. This will make significant progress towards software-defined security, and policy-driven autonomic cyberdefense.

5 Towards Secure and Dependable Service Function Chaining (SFC)

Our threat taxonomy established in Chapter 3 clearly indicates that a large set of novel threats will be introduced in NFV environment, among which the ones targeting at VNF layer are particularly destructive. In particular, one of the promising advantages offered by NFV and SDN is Service Function Chaining (SFC), which provides the network operators with the ability to integrates an ordered list of desired VNFs together for implementing a particular service to suite their specific needs. Although NFV/SDN based SFC brings significant agility, adaptability and flexibility, meanwhile reducing the hardware cost and operational complexity, its security and dependability issues have not been sufficiently studied. One significant issue that has been overlooked is the gap between high-level SFC specification and its enforcement at data plane. Once the high-level SFC policy is specified, there is no way to ensure that the VNFs are always chained in an expected manner, while the packet flows related to that service chain are forwarded correctly to the VNFs of concern in a predefined order. An attacker can manage to bypass or evade the security VNFs (*e.g.*, firewall, virus scanner, DPI) and deviate the packet flows from the pre-specified path. It is thus a significant need to have an efficient self-checking mechanism in place, ensuring the SFC to be implemented in a secure and correct way. As a result, this Chapter aims at developing a scheme based on an improved crypto primitive, called *lite identity-based ordered multisignature*, which enforces all the VNFs in the same service chain to sequentially sign the packets received. Then a verifier at the end of the chain will verify the aggregate signatures, so as to validate the authenticity of VNFs, as well as their orders in the chains.

5.1 Introduction

Service Function Chaining (SFC) [96, 148] (also known as VNF forwarding graph) refers to the capability of defining a set of service functions (*e.g.*, firewall, NAT, DPI), which are then stitched together in the network to create a service chain. The SFC capability can be used by network operators to set up arbitrary service chains from the instantiated VNFs to meet their application-specific requirements, while allowing to use a single network connectivity to chain many virtualized network functions as needed. Each service chain may serve specific functionalities that different from other service chains. For example, in data center use case, a service chain can be specified as SFC_i : NAT $\rightarrow$ firewall $\rightarrow$ virus scan $\rightarrow$ DPI, while video-based application use case the service chain can be defined as SFC_j : parental control $\rightarrow$ firewall $\rightarrow$ video optimizer (see Fig. 5.2 for more details). In practice, in order to steer traffic flows through an ordered list of service functions, SFC can leverage the capabilities of SDN

to obtain information gathering from network connection topology, service requirements, the availability of resources, and the current network status to determine the best forwarding path.

In order to orchestrate the underlying NFV infrastructure resources, initiate a set of VNF instances[1], and stitch them together in a logical ordered-fashion to create a service chain, two essential functional blocks must be put in place.

- First, NFV relies on *orchestration module* (a.k.a NFV MANO [74]), which plays a significant role in maintaining the full lifecycle management of network resources and services. Its functionalities include: (1) management of virtualized resource availability/allocation; (2) initiating, scaling or terminating the VNFs using the VNF on-boarding artifacts; (3) on-board network service, *i.e.*, register a network service in the catalog and ensure that all templates describing the network service are on-boarded; (4) creating, updating and deleting network servic and the VNF forwarding graph associated to the network service.
- Second, it leverages *SDN controller module* [70, 165] to provide the full network control capabilities such as traffic steering and forwarding, routing decision based on the global view of network status, and path provisioning between the instantiated VNFs. In particular, SDN is recognized as the complimentary technology to NFV, and it has been practically used as a part of SFC. Thanks to the SDN controller's capabilities to dynamically manage VNFs virtual connections and underlying data plane flows, the network operators can reconstruct and update a particular service chain and steer the traffic in a flexible and automated way.

Recently, there are several ongoing research efforts which attempt to address the unique and unprecedented challenges imposed by SFC. However, we observe that consistency problem in SFC is one of the major concerns and remains comparatively overlooked by the research community. The challenge arising here is that once the high-level SFC policy is specified, how we can make sure that it is correctly translated into the network flow classification with accurate packet forwarding rules, or how to ensure and prove that the packet flows associated to a particular service chain are traversed correctly to appropriate and legitimate VNFs with respect to the predefined policy, achieving both security (*e.g.*, authenticity, integrity of VNFs) and dependability (ordering property) of the service chain in NFV environment. Anomalous flow redirection and path deviation [205, 247] are good examples of attack model that can be used by the attackers to deviate the original service function path. Their objective is to bypass or evade from security functions in the service chain, ultimately leading to the violation of SFC policy. In other words, to achieve the goal, they can either launch rule modification attack [247, 132, 35] against victim switches at SDN data plane or topology tampering attack [105, 59, 214] at SDN control plane.

- Considering the data plane, assume that if the SFC policy defined the packet flows must go through a firewall, and if the attackers have successfully performed rule modification attack [247, 132, 35] on the victim switch to subvert or modify the original rules installed by the SDN controller. Thus, all packets belonging to this flow will bypass or evade from the firewall (*i.e.*, security function in a service chain). Note that the use of SDN by itself provides no means to detect path forwarding anomalies. In fact, the controller only installs a set of forwarding rules on the data plane switches, but it cannot ensure that whether the installed rules are correctly enforced with the benign forwarding behaviors.
- At the control plane, the attackers may launch topology tampering attack [105, 59, 214] to poison the controller's view of the network topology information, and deceive the controller to trust a spoofed topology. To do that, they may perform DDoS attack on the victim

[1] In the paper, VNFs and service functions (SF) are two interchangeable word. The VNF's terminology is defined by ETSI, while service functions specified by IETF.

host/virtual machine/VNF to make it unavailable and try to impersonate this victim node claiming that it has already migrated and moved to the new location. This situation creates a new update/change in the network connectivity, and causes a mismatched location information between the Host Profile[2] and Packet-In message received from the data plane switch. As a result, the controller assumes that the host has moved to a new location, it then updates the location information inside the corresponding Host Profile. Ultimately leading to the false in flow rule installation over the switches, and resulting in deviating the traffic flow from victim's actual location to the attacker's location. However, such update mechanism is vulnerable due to the ignorance of host authenticity.

Since SFC is still in its early stage, from security point of view, it is significant and urgent to develop a new security primitive to achieve both security (*i.e.*, authenticity, integrity of VNFs) and dependability (*i.e.*ordering property), in addition to optimality, of the service chain.

5.2 Related Work

SFC is not an entirely new concept but is has become an active research domain since the emergence of NFV and SDN. Although the concept of *"service function chain"* has been proposed long time ago, it is very difficult to be implemented due to the limitations of traditional network. Thanks to the promising benefits of NFV and SDN, SFC finds its interesting use case, and becomes an active research direction. Nevertheless, the initial concept of SFC was raised by IETF SFC working group to develop its architecture and define how network flow classification can be used to route traffic between service functions [96]. However, this traditional model does not explicitly commit to support SDN and NFV [69]. As a result, ETSI NFV working group has been motivated to integrate this SFC functionalities (as defined by IETF) into ETSI NFV architectural framework [69], and give a new terminology called VNF forwarding graph (VNFFG)[3]. Nowadays, there are several ongoing research works attempting to address the unique and unprecedented challenges imposed by SFC. Some of them are focused on implementation aspects (*e.g.*, [186, 249, 50, 148, 232, 54]), while others focus on optimization theory, models and provable analysis of algorithms to achieve an optimal deployment of SFC (*e.g.*, [12, 137, 63, 208, 90, 91, 147, 108, 14]). Nevertheless, there are very few studies dedicated towards security and dependability concerns related to SFC. The closely related work falls into two general categories,

1. **Static verification:** There are only a few proposals targeting static based verification, some examples are [86, 228, 251]. Overall objective is to provide a static verification mechanism for checking the correctness of forwarding behaviors in SFC. ChainGuard [86] makes use of SFC-related rules that are stored within the flow tables of virtual switches to gather the actual SFC Overlay and Traffic Steering (SOTS) snapshot and model it as property-based graphs. These graphs are then used to examine whether the actual SOTS is conformed to the required SFC specification. SFC-Checker [228], which contains a Stateful Forwarding Graph (SFG) representing how packets are forwarded, how the states of service function are changed, and how the state changes affect the forwarding path. Next, a graph traversal algorithm is applied to identify the paths that satisfy the queries (*e.g.*, under what scenarios, can all A's packets reach B?), and aided network operators in their SFC diagnosis. A Quantitive Forwarding Graph (QFG) has been proposed in [251],

[2] Inside the SDN controller, the Host Profile is maintained to track the location of a host.

[3] Indeed, VNFFG and SFC provide the same definition. It describes the order of VNFs in a service chain. The single VNFFG contains one or more Network Forwarding Paths (NFPs). In the SFC terminology, the VNFFG is considered as the SFC, VNFs are equivalent to SFs, NFPs are SFPs, policies defined by network service provider are the rules used for the SFC classifier, and Virtual Links (VLs) are implemented by one or various SF Forwarders.

which built on top of exiting work on the SFG [228] to extend the capability of SLA verification, *i.e.*, whether the network in a given configuration can deliver the performance within the SLA bounds.

After a careful study on their approaches, we observed that they introduce two major drawbacks. First, it is a static verification. The verification has been examined based on the forwarding graph that indeed generated according to the network information such as topology, flow tables, service states, *etc.*. According to this approach, it incurs a large number of overhead hence a verifier must periodically pull the underlying information from the data plane devices (*e.g.*, the flow tables from the virtual switches), and regularly regenerate the forwarding graph when a change occurs. Second, it is not trivial to extract the right information from a large number of flow tables entries which are relevant for SFC verification. However, the aforementioned works do not provide a clear explanation how the relevant parts of SFC verification are gathered and transferred into the verification model.

2. **Active verification:** SFC Path Tracer [65] is a tool for troubleshooting SFC. Initially, the controller artificially injects probe packet in the chain input to generate the trace. The probe packet is flagged by Explicit Congestion Notification (ECN) field in the IP header [195] with two bits. Once the probe packet traverses the network elements in the target chain (*i.e.*, when it leaves a forwarder switch to its next hop), it is mirrored by the trace tool to discover which forwarder handled the packet. SDNsec [205] and REV [247] are other two active verification mechanisms, in which each switch along the forwarding path computes a Message Authentication Code (MAC) and attaches as a tag to each packet. In SDNsec, MAC is computed with the shared key sharing between the controller and the corresponding switches on the path. The controller can instruct any switches to provide feedback/information and thus inspect the path that was taken through analyzing the tag. Similarly to SDNsec, the REV relies on the same idea of using a symmetric key to compute MAC. At the end, a destination switch leverages a public/private key to generate signature, attaches its with a verification report and sends back to the controller for further examination. One problem of REV is that it comes at the cost of complicated key management and has a high packet overhead, because it implemented based on RSA primitive. Although SDNsec is attempted to reduce the key management cost thereby lightening security property to use only symmetric key for MAC computation (*i.e.*, 128-bits AES), it still incurs engineering complexity to modify and eventually add specific forwarding information (such as forwarding entry field, path validation field, egress switch ID, flow ID, *etc.*) into the packet header fields. The same complexity issue also occurs in SFC Path Tracer, hence it required to set ECN bits in the IP header to trigger OpenFlow rules and installed the related trace rules in switch tables, such instructing the switches to copy those mirrored packets to the trace tool. It is worth nothing that, this approach generates heavy traffic overhead between the controller and switches.

5.3 Background and Challenges

In this Section, we firstly explain the background of SFC, its relationships with SDN and NFV MANO functional block in ETSI NFV architectural framework, highlight some opportunities offered by SFC, and then identify the potential challenges.

5.3.1 SFC working principles

5.3.1.1 Architectural description

The implement of SFC involves several steps, from defining high-level network service description to resource allocation, VNF instantiation and placement, to VNF selection, SFC composition, traffic steering and forwarding. For a better understanding about SFC architecture and its relationship with SDN and NFV, we start with explaining the architecture model from high-level business description to low-level service deployment.

As presented in Fig. 5.1, the network operators specify high-level description about network service, such as application topology, network policy, and SFC policy, which instruct the traffic flows of a particular user or tenant to go through an appropriate set of ordered service functions from a given source to destination. Nowadays, different data modeling languages have been used to describe the deployment and operational behavior requirement towards constructing an end-to-end network service (*a.k.a*, a service chain). Some are de-facto standards and well-known, while others are developed from scratch. The most notable language and widely used in today's NFV orchestration is TOSCA (Topology and Orchestration Specification for Cloud Application) language [227]. It is introduced as a standard approach to defining NFV specific data models and deployment templates which contain necessary information used by the NFV orchestrator for life cycle management of network services. In a nutshell, the TOSCA-based service descriptions have been shared by several components in NFV architecture. NFV orchestrator parses the information related to VNF descriptions to Virtualized Network Function Manager (VNFM) for instantiating specific VNF instances (*e.g.*, virus scan, firewall, DPI), and the corresponding information to VIM for allocating NFVI resources (*e.g.*, compute, storage, and network). The SDN controller is used to receive all necessary instructions from the VIM, managed the underlying networks and configured the flow rules. The main functionalities include: (1) managing a data-plane elements including classifier, Service Function Forwarder (SFF), VNFs, and proxy devices; (2) providing the SFC data path programmability, and (3) instructing flow rules to the classifier, SFF and proxy devices to direct the flows to VNFs based on the specified SFC request.

Specifically, classifier[4] acts as entry and exit points for SFC-based traffic steering. It performs packet classification, provides SFC encapsulation, and directs the matched traffic to appropriate service function paths. To encapsulate the packets, it adds SFC-header which contains Service Path Identifier (SPI) and Service Index (SI) to each packet. Specifically, the SPI represents the service path for a particular SFC and identifies the ordering (position) of VNFs that must be performed, while SI is used to determine the next VNF to be traversed [188]. Once the packet reaches the SFF, it removes the outer encapsulation and trigger a lookup based on SPI and SI to identify the outgoing encapsulation. For example, the packets are either forwarded to SFFs or VNFs. Meanwhile, VNF is responsible for specific treatment of the received packets. It can be either NSH-aware (Network Service Header) or NSH-unaware VNFs. In case of NSH-unaware VNF, an additional proxy function is added and interfaced with this NSH-unaware VNF, with the objective to perform SFC-deencapsulation before forwarding the packet to the NSH-unaware VNFs. Then it performs SFC-reencapsulation before sending the packet to the SFF. For simplicity of use, we assume that those deployed VNF instances are NSH-aware.

5.3.1.2 Salient features and major advantages

In the following, we briefly highlight the major advantages introduced by SFC.

[4] In SFC-enabled domain, classifier can be classified into two types: ingress classifier (or SFC ingress node) and egress classifier (or SFC egress node). Both ingress and egress classifiers denote as the SFC boundary nodes. The ingress node aims at handling traffic entering the SFC-enabled domain, which responsible for performing the SFC-encapsulation to the packet. While, the egress node handles traffic leaving the SFC-enabled domain. It is required to remove any information specific to SFC domain (*a.k.a*, SFC-deencapsulation).

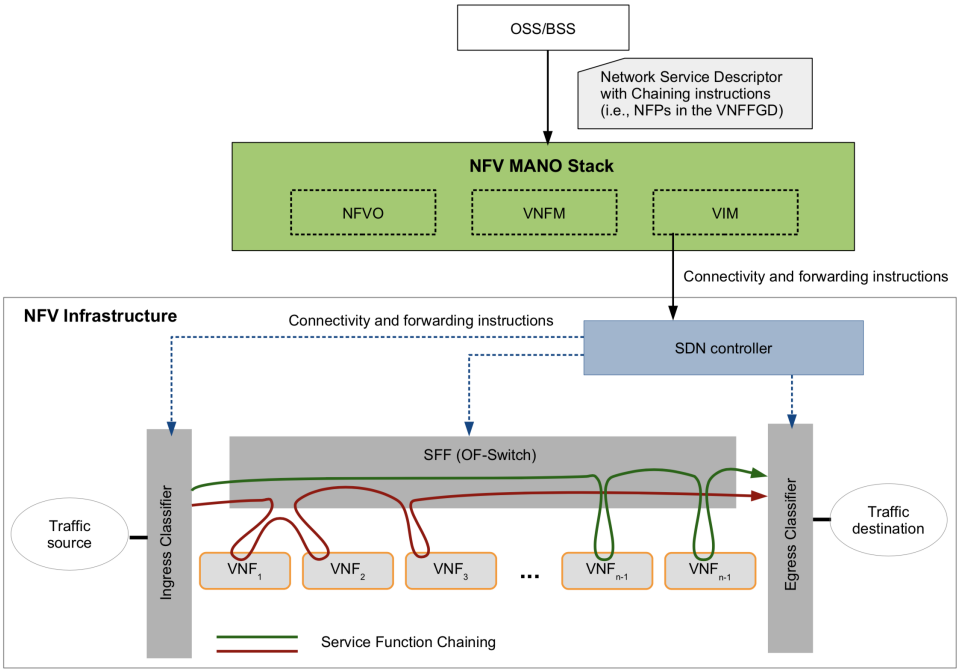


Figure 5.1: Intra-domain orchestrated architecture with the corresponding SFC workflow from high-level business specification to actual SFC deployment enabled by NFV and SDN

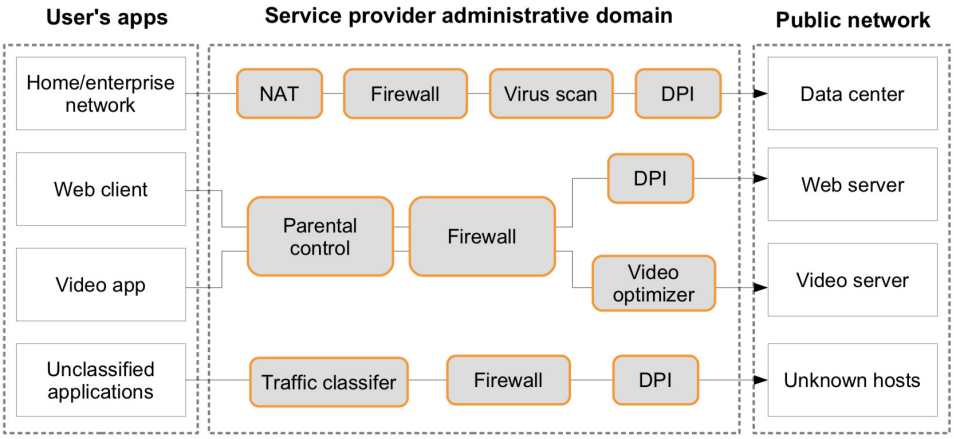


Figure 5.2: Motivating examples of SFCs

- **On-demand VNF chaining.** With the quickly growing demand for cloud services and the numerous business-specific requirements, different service providers tend to have the capabilities of customizing their own network services to meet the particular needs of users and tenants. In traditional networking model, it appears technical difficulties for network operators to create such a service chain of composing different types of VNFs. Also, it is hard to steer the traffic flows to go through the sequences of designated service functions at a granularity level. For example, the traffic flows coming from the different subscribers may enforce by specific service chains (*e.g.*, video streaming, security control, or monitoring purposes) which depend on their behaviors and characteristics. Thanks to NFV and SDN, network operators can arbitrary create different service chains to meet user specific requirements. For example, to provide security control, the network operators can customize a service chain as SFC_x: firewall $\rightarrow$ virus scan $\rightarrow$ DPI. Similarly, to improve the performance of website, they can define the service chain as SFC_y: web proxy $\rightarrow$ load balancing, and so on.

- **Simplified SFC provisioning.** In the past, building a service chain of composing different service functions to meet user specific needs tend to be a complex task, non-trivial administrative burden, error prone and time consuming effort. Due to the fact that network elements are developed based on specialized hardware devices, each of them had to be individually configured with its own command syntax. They are linked together by a complex and rigid set of configuration to create a service chain as well as a service function path. Thus, the result of static chain configuration cannot flexibly cope with dynamic changes in user requirements, and rapid service migration and provisioning in the clouds. For example, dynamic insertion of new service functions, scaling out the number of services for load balancing. However, this limitation can be overcome by leveraging NFV/SDN. As a matter of fact, the benefits of SFC brings more flexible and elastic solutions, while providing network operators the capability to customize their own software-based service chain on the fly with no longer requiring specialized hardwares and manual intervention to configure the network.

- **Resource sharing across multiple service chains.** As aforementioned, with the complex configuration, physical hardware constraints and vendor specific properties, the network operators are lock-in with the management capabilities in providing efficient SFC deployment. For example, suppose vendor X is better at serving application layer firewall (firewall-L7), but vendor Y is doing good for network layer firewall (firewall-L3-L4). Due to the proprietary nature of service components, un-unified standard or even physical constraints, this may force network operators to pick only one hardware appliance. In contrast, with the NFV/SDN-based SFC capabilities, vendor locked-in problem can be tackled. We can avoid the undesirable situation of choosing from one vendor. Hence the service functions are virtualized running in a virtual machine as a VNF instance, so that network operators can quickly instantiate various service functions and dynamically insert them into a specific SFC to meet their particular needs, *i.e.*, security, load balancing, or QoS/flow control purposes. Another benefit is that a common and widely used service functions such as firewall, DPI, *etc.*, can be shared and leveraged by multiple service chains. The motivating examples are given in Fig. 5.2, in which a particular SFC is applied based on different traffic flows depending on the source, destination or type of traffic. In addition, the common and the most widely used service functions can be shared between them.

5.3.2 Challenges

Although SFC brings significant advantages to service providers and network operators (as aforementioned), it is currently still a concept-level technology and many challenges need to be tackled before moving towards practical and commercial uses. Below we intend to articulate some security challenges associated with SFCs.

- **Reliable and consistent VNF chaining:** The ability to build a complex or a specific service chain by composing several types of service functions and stitching them together in the network is one of the key features offered by SFC. However, a newly proposed NFV/SDN-enabled SFC can be vulnerable to new pitfalls which can be exploited by the attackers. For example, they may target individual SFF using rule modification attack to tamper the flow rules [247, 132, 35]. As a result, packets can deviate from their original service function path, ultimately violating original SFC policies. The main reason that make this issue very important, considering to the fact that if a rule has already been defined traffic flow to go through some specific service functions. Let's say it must respectively pass through firewall and intrusion detection services. Imagine that if the attacker

can modify/delete this rule, thus all packet belonging to this flow will bypass or evade from these specified security functions without being detected. However, our study reveals that there is a great challenge to examine the actual SFC deployment at the data plane whether it is secure, reliable and consistent chaining with a specified set of VNFs and truly conformed to the predefined SFC policy. While, the packet associated to that service chain is absolutely traversed to all intended service functions.

- **Ordering property preservation.** In particular, end-to-end application traffic flows are often required to traverse various VNF instances in a sequence manner as specified in SFC policy. For example, a user flow accessing an application server at a remote site may need to go through (1) WAN optimizer to accelerate application delivery, (2) firewall and DPI for security control, (3) load balancing to maximize speed and ensuring optimum utilization of available resources, and (4) a VPN server before reaching the application server. One of the key observations is that these VNF instances involved in a particular service chain have been deployed independently. Each VNF is responsible for specific treatment of received packets, and totally unrelated to other service functions [189]. That says, VNF_1, VNF_2, ..., VNF_n are unrelated, and there is no notion or representation at service layer that VNF_1 should occur before VNF_2, and so on. Although many service functions in a service chain are clearly defined in a strict order, the challenge arising here is that network operators have no consistent way on how to verify or validate the ordering property of VNFs in their actual SFC deployment. For example, whether they have been chained correctly in a sequence manner. Out-of-order traversal attack [247, 125] is a good example of attack that can substantially disrupt the ordering (position) of VNFs in the service chain, ultimately resulting in undependability and inconsistency. To do that, attacker may deviate traffic flow to not traverse on the intended forwarding path with the right order. Clearly, assume that a SFC policy has been defined a particular flow to go through firewall and later encryption. If the attacker can modify the rules by deviating the flow to firstly pass through encryption before firewall. This makes the hidden threats in the encrypted traffic can be evaded and incapable of detecting by the firewall. So far, our study reveals that the verification mechanism for examining the correct sequence of service functions (or correct packet traversal) in a service chain has not been extensively studied.

- **VNF authenticity.** Verifying the authenticity of service functions has not yet been addressed in much detail so far, especially in the context of SFC. One of the great challenges is that the network operators may not even notice and having the right to examine whether a set of VNFs involved in their service chains is compromised, or whether there exist any malicious service function instances. An example attack scenario is that the adversaries may launch topology tampering hijacking attack [105, 59] or DDoS attack [214] in order to make the legitimate VNFs unavailable and then try to impersonate these victim nodes thereby receiving and responding the packets destined to them. This situation leads to the network nodes including controller and other VNF instances could be communicated and synchronized with the adversary instead of the legitimate ones. As a result, without sufficient knowledge and adequate authenticity protection, this can potentially break down the integrity of individual service function towards impacting the reliability, consistency and dependability of the whole service chain.

5.3.3 Contributions

To tackle the above challenges, we have made the following contributions in this Chapter,

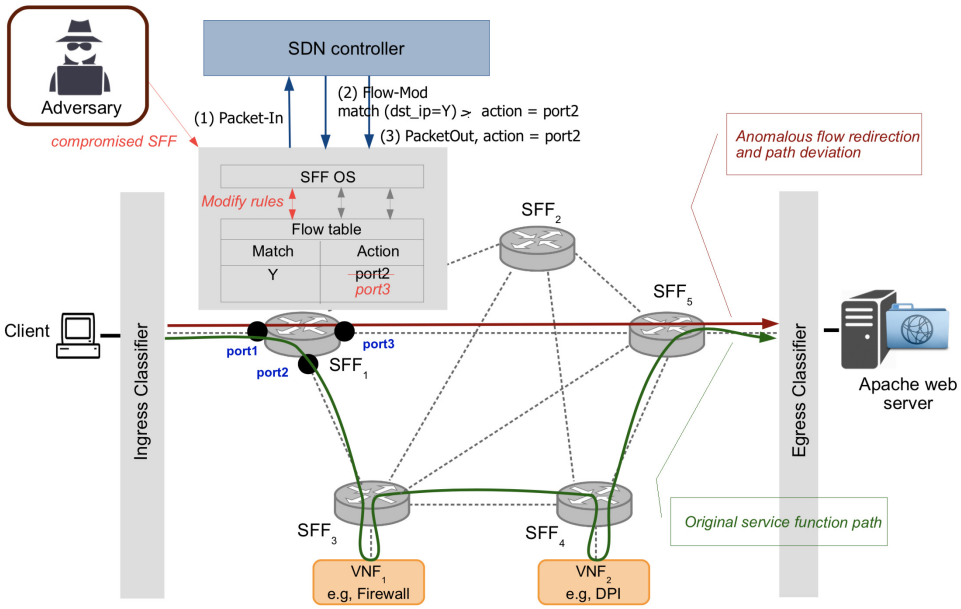


Figure 5.3: Illustration of rule installation in SFF and its attack model

- We propose a new security primitive, called *Lite identity-based ordered multisignature* which provides efficient verification mechanism to examine the behavior of packet traversal and verify whether all the packet flows associated with a particular service chain have been handled correctly as specified in SFC policy.

- We give security analysis on our scheme and demonstrate that it preserves five properties simultaneously, (1) *Unforgeability*, which is computationally infeasible for any adversary to produce an aggregate forgery implicating honest identities; (2) *Authenticity*, ensures that only honest VNFs can produce a valid signature; (3) Re-order protection, it is not possible to re-oder the positions of honest VNF involved in a service chain; (4) *Constant-sizes* in keys and aggregate signature; and (5) Signature and verification acceleration using three-pairing computations in bilinear maps. We provide sound theoretical proof and show that the scheme can prevent many attacks like anomalous flow redirection and path deviation.

- A real-world platform and experimental testbed is established for implementing our scheme and examine its feasibility in real-life settings. The implementation codes will be shared with the NSH community and contribute to its further development. We then develop a prototype of our scheme and evaluate the performance of Lite identity-based ordered multisignature.

5.4 Problem Statement

As aforementioned, SFC provides the ability to define an ordered list of network functions (*e.g.*, firewall $\rightarrow$ virus scan $\rightarrow$ DPI), creates the desired service chain, and steers traffic flows through the predefined service function path. Although SFC offers significant benefits, it inevitably comes together with the new threat vectors, *i.e.*, reliability and consistency problems, which have not been extensively addressed. Therefore, this section aims at describing system model, identifying the threat models and finally highlighting our key ideas to address these threats.

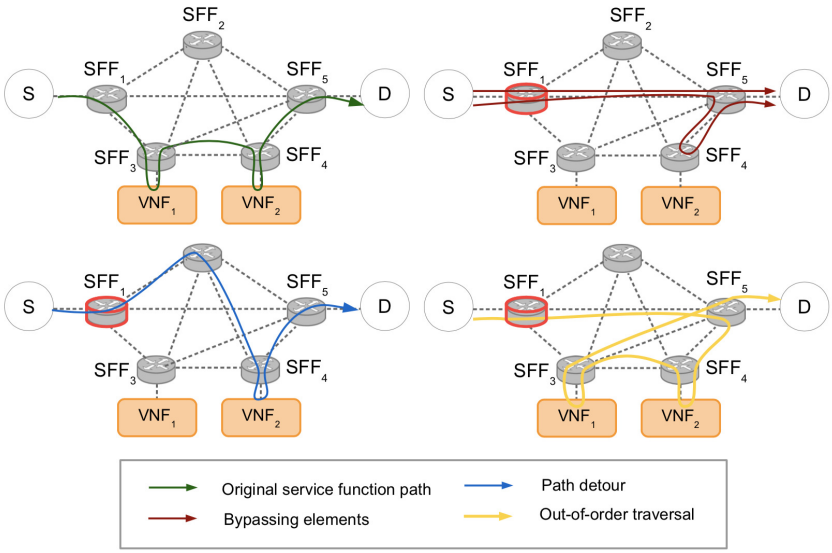


Figure 5.4: Forms of anomalous flow redirection and path deviation

5.4.1 System model

For concreteness, we give more attention to the underlying data plane operations for setting up traffic steering and constructing service function paths. Typically, in each intra-domain orchestrated architecture, there is a centralized controller which manages and controls a set of data-plane elements (*e.g.*, classifier, SFFS). Network operators can either specify high-level SFC policies using Network Service Descriptor (NSD)[5], or directly interact with SDN controller through the provided SFC API. The controller translates SFC policies into a specific service function path, adjusts the path based on VNF status and overlay links, installs and governs a set of forwarding rules on the data-plane elements accordingly.

To steer the traffic flows, the data-plane elements like SFF utilizes the flow rules installed in its flow tables to determine the network forwarding path and decide which action needs to be performed. In particular, the flow rules consist of two parts: (1) *match fields* which filters packet headers; and (2) *instructions* indicating what actions need to be taken when the matched packets are found, *e.g.*, drop packet, forwarding to the port. Upon arrival of new packet, SFF checks if the packet matches any existing flow rules. If so, it processes the packet based on the matching rule with the highest priority. Otherwise, the SFF sends a Packet-In message to the controller to ask for proper actions. The controller decides on the route of packet and sends the corresponding rules to the SFF through a standard control channel like OpenFlow [164]. This event is known as Flow-Mod messages. Fig. 5.3 illustrates an example where a routing action is taken once a matched packet with destination Y (*e.g.*, Apache web server) and source X (*e.g.*, client) arrived at $SFF1$.

5.4.2 Threat model

This Section presents several important threat models that potentially lead to anomalous flow redirection or path deviation. Specifically, those attack models can compromise both security (*e.g.*, authenticity, integrity of VNFs) and dependability (*e.g.*, ordering property of the service chain) of SFC.

[5]According to ETSI NFV document [71], NFV orchestrator uses NSD (which is a deployment template) to represent high-level logical network construction. In particular, NSD consists of information used by NFV orchestrator for lifecycle management of a network service including defining the service chain, deploying VNF instances, creating virtual links and service function path accordingly.

The basic idea of anomalous flow redirection and path deviation is to deviate the packet flows from their intended forwarding paths that may violate SFC policies. We generally consider two families of attacks,

- **Rule modification attack** [247, 132, 35], which target at the victim SFFs in order to deviate the original service function path. Once it succeeds, they can further manipulate or modify the SFF's resident flow rules to enforce anomalous packet redirection. Specifically, it can be further classified into three different forms (as shown in Fig. 5.4):
  - *Bypass elements*, in which one or more VNF appliances on the intended service function path have actually been bypassed or skipped;
  - *Path detour*, the path deviates from the original path but later the packets will return to the correct next-hop downstream VNFs; and
  - *Out-of-order traversal*, an adversary deviates the intended service function path to not traverse in the right order as specified in the SFC requirement.

  To achieve these goals, adversaries can compromise the SFF by exploiting vulnerabilities to SFF's OS and install backdoor applications on it for allowing them to arbitrary perform malicious operations such as install, delete, or modify the flow tables. As exemplified in Fig. 5.3, they may modify output port of forwarding rule installed at the flow table of the compromised $SFF1$ in order to evade from security functions (such as firewall and DPI) involved in a particular service chain.

- **Topology tampering attack** [105, 59, 214], in which attackers intend to spoof the controller's view of the network topology, and deceive it to believe in a spurious topology. This causes the false in flow rule installation over the SFFs, resulting in redirecting the traffic flow from victim's actual location to the attacker's location. As inside the controller, a global view of network topology including hosts/virtual machines/VNFs, SFFs, and the details of link connections have been maintained by the Host Tracking Service (HTS) and the Link Discovery Service (LDS) respectively. The controller reactively listens to Packet-In messages coming from SFFs to maintain the Host Profile. In this attack scenario, we assume the attacker can launch host location hijacking to corrupt HTS module by spoofing the victim's address information (*e.g.*, VNF's IP address). This leads to the mismatched result of location information between the existing Host Profile and the information received from incoming Packet-In messages. According to this event, the HTS makes an assumption that the host has been migrated and moved to a new location (it is, however, not true), and sequentially updates the new location information inside the corresponding Host Profile. This attack is due to the ignorance of host authenticity checks.

In addition, we assume the controller itself is trustworthy (which required for the correct functioning of the network) and the transmission of message via control channel can be properly protected using SSL/TLS. However, the inverse message transmission from SFFs to the controller is untrusted and could be forged by malicious SFFs.

5.5 Proposed Solution

To achieve secure, consistent and dependable SFC against anomalous flow redirection and path deviation, we propose a new security primitive, namely *Lite identity-based ordered multisignature* scheme. The objective is to enforce each VNF involved in a service chain to attest its signature on the packet received, so that a verifier can verify whether the service chain has met

certain criteria, *i.e.*, VNF chaining/composition, ordering and authenticity properties, by validating the received signature. Specifically, our proposed scheme is motivated and derived from a combination of two signature schemes (ordered multisignatures and identity-based sequential aggregate signatures) proposed in [22]. Therefore, we begin the section by describing the basic ideas of these two schemes, followed by deficiencies analysis in terms of signature construction and verification. Finally, we highlight how our proposed scheme of Lite identity-based ordered multisignature can deal with these deficiencies.

5.5.1 Preliminaries

As aforementioned, Lite identity-based ordered multisignatures scheme is underpinned and motivated by two key principles: (1) *Ordered multisignatures* which allow signers *i.e.*, VNF appliances, to attest their aggregated signatures in which they signed (along the service function path) by respecting the ordering of signers; (2) *Identity-based sequential aggregate signatures*, instead of relying on a public-key primitive, *i.e.*, RSA cryptosystem, each signer can use an arbitrary string (*e.g.*, an IP address) to act as a user's public key and thus verifying signature needs only knowledge of a sender's identity, without requiring knowledge of traditional public keys. As a result, the setup and storage overhead of obtaining and storing these public keys can be reduced.

To have a better understanding, Algorithm 1 describes step-by-step procedure of creating ordered multisignatures. It typically relies on four algorithms.

- *Global parameter generation (OP):* that outputs some global information I for the scheme. This algorithm can be run by a trusted third party or standard bodies.
- *Key generation (KeyGen):* run by a user on the inputs I , the algorithm returns a public-secret key-pair (pk, sk).
- *Signing (Sign):* the algorithm first asks for help from verification algorithm $\mathsf{Vf}(m, \sigma_i, L_i)$ to verify the consistency of an aggregate signature received from the previous participants. If it is valid, then it takes inputs (sk_i, m, σ_i, L_i) and returns a new aggregate signature $\sigma_{i'}$. Here, sk_i is a valid secret key, $m \in \{0,1\}^*$ is a message, σ_i indicates an aggregate signature obtained from the previous participants, and $L_i = (pk_1, ..., pk_i)$ is a list of corresponding public keys.
- *Verification (Vf):* which takes as input a tuple (m, σ_j, L_j), and returns a bit 1 if the consistency of aggregate signature is retained, otherwise returns 0.

Algorithm 1 Ordered multisignatures algorithm

$I \xleftarrow{\$} \mathsf{OP}$ ▷ \$ means uniformly selected at random
$(pk_1, sk_1), ..., (pk_n, sk_n) \xleftarrow{\$} \mathsf{KeyGen}(I)$
$\sigma_0, L_0 \leftarrow \varepsilon$ ▷ ε indicates empty string
for $i = 1, ..., n$ **do**
    $\sigma_i \xleftarrow{\$} \mathsf{Sign}(sk_i, m, \sigma_{i-1}, L_{i-1})$
    $L_i \leftarrow (pk_1, ..., pk_i)$
end for

Algorithm 2 is more closely related to identity-based sequential aggregate signatures scheme. Particularly, it has been designed with the purpose to reduce setup and storage overhead occur in the above ordered multisignature scheme with associated to distributing public keys and corresponding certificates, and having participants to store the keys indefinitely. Similar to

the above algorithm, identity-based sequential aggregate signatures scheme contains four algorithms. There are a few differences on key derivation and signing processes as discussed in the following.

Algorithm 2 Identity-based sequential aggregate signatures algorithm

1: $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$
2: **for** $i = 1, ..., n$ **do**
3: $\quad sk_{id_i} \xleftarrow{\$} \mathsf{KeyDer}(msk, id_i)$
4: **end for**
5: $\sigma_0, L_0 \leftarrow \varepsilon$
6: **for** $i = 1, ..., n$ **do**
7: $\quad \sigma_i \xleftarrow{\$} \mathsf{Sign}(sk_{id_i}, m_i, \sigma_{i-1}, L_{i-1})$
8: $\quad L_i \leftarrow ((id_1, m_1), ..., (id_i, m_i))$
9: **end for**

- *Setup algorithm (Setup):* initially runs by the trusted Private Key Generator (PKG) to generate its master public key (mpk) and the corresponding master secret key (msk).
- *User's secret key derivation (KeyDer):* run by the PKG on inputs (msk, id), the algorithm generates the corresponding user's secret key sk_{id} for any user identity $id \in \{0, 1\}^*$.
- *Signing algorithm (Sign):* before signing, the algorithm first checks the consistency of received aggregate signature using the help from verification algorithm $\mathsf{Vf}(mpk, \sigma_j, L_j)$. If the condition is met, it continuously computes a new aggregate signature on inputs $(sk_{id_i}, m, \sigma_i, L_i)$; where $L_i = ((ID_1, m_1), ..., (ID_i, m_i))$ is a list of identity-message pairs.
- *Verification algorithm (Vf):* that takes inputs (mpk, σ_j, L_j) and returns a bit 1 if the aggregate signature is valid, otherwise returns 0.

Discussion. The above schemes bring two significant enhancements in terms of constant sizes (in that both secret keys and aggregate signatures) and signing order preservation. Thus, it is not surprising to see that several network routing applications are widely applied these schemes to secure route attestation. That's means, a data packet should be signed in sequence manner by egress routers, allowing ingress routers to accept and forward only packet that followed an authenticated path, and finally the originating source can later verify whether the packet actually took an authenticated path to reach its destination [22]. However, the verification process in the above schemes inevitably introduces high computational overhead. Due to the fact that each intermediate node needs to firstly verify the validity of aggregate signature received from the previous nodes on the path before starting the signature construction. Imagine that if the number of nodes on the path increase linearly, this would significantly impact the performance of signature verification. Another security concern is that the authors of [111] found that the signature construction in the above schemes are insecure and universally forgeable, that is anyone can generate forged signatures on any messages of its choice.

5.5.2 Design properties

Although the above schemes seem to be suitable for use in SFC domain over the NFV environment and can be applied to tackle the SFC consistency challenges, *i.e.*, VNF chaining/composition, ordering and authenticity properties (as discussed in Section 5.3.2), their models still vulnerable to forgery attack while the computational overhead from signature verification is very high. Therefore, Lite identity-based ordered multisignature is proposed with

the objective to (1) improve security in signature construction to prevent against forgery attack, (2) accelerate computation times by using three pairing computations, and (3) eliminate the verification process in the intermediate nodes. Meanwhile, other common key features derived from the above schemes have also retained. For concreteness, Lite identity-based ordered multisignature is encompassed by the following properties.

- *Unforgeability:* by adapting security model for aggregate signatures [23, 15], it should be computationally infeasible for any adversary to produce a forged aggregate signature implicating an honest identity, even when the adversary can control all other identities involved in the aggregate signature and can mount a chosen-message attack on the honest identity.

- *Authenticity:* signer's authenticity should be preserved. Given a message, the corresponding aggregate signature does not only provide the knowledge indicating that some specific group of signers (*e.g.*, VNF appliances) signed it, but also to the order in which they signed. So that a verifier can later verify the authenticity upon receiving the aggregate signature.

- *Re-order protection:* it must enforce an additional unforgeability with respect to the ordering of the signers. In other words, it should not be possible to re-order the positions of honest signers, even if all other signers are malicious.

- *Constant size:* the sizes of aggregate signature at any stage should be constant regardless the number of signers and messages.

- *Signature and verification accelerations:* an alternative solution to minimize overall computational overhead is to ignore verification call at intermediate nodes and accelerate the construction time using three pairing computations. Unlike the network routing applications which require more secure route attestation, our objective aims to verify the dependability and consistency for the entire SFC, so that there is no need to perform intensive verification before signing. While three pairing computations in the terms of (X, Y, Z) can help to accelerate the construction times in both signature and verification. Such each term is executed in parallel, resulting in a much faster computation time when compared to a typical model-based one-time processing. See section 5.5.3 for more construction details.

5.5.3 Construction methodology

Our construction uses bilinear maps (*a.k.a*, pairings) on groups of prime order. Let $\mathbb{G}, \mathbb{G}_T$ be the groups of the same prime order p. Based on a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, it holds two following properties [23]. All the notations used in our formulation are summarized in Table 5.1.

- *Bilinearity:* for all $(u, v) \in \mathbb{G}$ and $(a, b) \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.

- *Non-degeneracy:* if $g \in \mathbb{G}^*$ is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$.

Definition 5.5.1. We call an algorithm $\mathcal{G}$ as a bilinear-group generation algorithm that returns outputs $(p, \mathbb{G}, \mathbb{G}_T, e)$.

To construct the Lite identity-based ordered multisignature, there are four steps involved.

Table 5.1: Notations used in Lite identity-based ordered multisignature formulation

Notations	Description
$e(.,.)$	A symmetric bilinear form. For example, $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$
$\mathbb{Z}$	A set of integers
$\mathbb{N}$	The positive integers
$\mathbb{Z}_p$	The integers modulo $p \geq 2$
$\mathbb{Z}_p^*$	The multiplicative group modulo p
$\mathbb{G}, \mathbb{G}_T$	Groups of prime-order p on rational points of an elliptic curve over a finite field
$1_{\mathbb{G}}, 1_{\mathbb{G}_T}$	The identity elements of $\mathbb{G}, \mathbb{G}_T$ respectively
$\mathbb{G}^*$	A set of generators of $\mathbb{G}$, *i.e.*, $\mathbb{G}^* = \mathbb{G} - 1_{\mathbb{G}}$
$\{0,1\}^*$	A set of all binary strings of finite length
ε	An empty string
$\|x\|$	If x is a string then $\|x\|$ is its length in bits
$x\|\|y$	If x, y are strings then $x\|\|y$ denotes an encoding from which x and y are uniquely recoverable
$s \xleftarrow{\$} S$	If S is a finite set then $s \xleftarrow{\$} S$ means that s is selected uniformly at random from S. For example, $s_1, s_2, ..., s_l \xleftarrow{\$} S$ as shorthand for $s_1 \xleftarrow{\$} S; s_2 \xleftarrow{\$} S; ..., s_l \xleftarrow{\$} S$; for any $l \in \mathbb{N}$
$x \xleftarrow{\$} A(y,z)$	If A is a randomized algorithm then $x \xleftarrow{\$} A(y,z)$ means that x is the output after being run A on inputs y, z. If A is deterministic then we drop \$ sign above the arrow.

5.5.3.1 Setup

The algorithm first run $\mathcal{G}$ to obtain outputs $(p, \mathbb{G}, \mathbb{G}_T, e)$. It then chooses a random generator $g \in \mathbb{G}^*$, a random number $\alpha \in \mathbb{Z}_p$, and two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ as random oracles. It returns $(p, \mathbb{G}, \mathbb{G}_T, e, g, g^\alpha, H_1, H_2)$ as the public parameters (PP), and α as the master secret key (msk); where g^α is the master public key (mpk). Please note that the PP and mpk have no much difference in essential. Both parameters are shared by all users in the system, so that we can view mpk as PP and vice versa.

Definition 5.5.2. We define $\mathbb{G}, \mathbb{G}_T$ are groups of prime-order p, and $1_{\mathbb{G}}, 1_{\mathbb{G}_T}$ denote the identity elements of $\mathbb{G}, \mathbb{G}_T$ respectively. By $\mathbb{G}^* = \mathbb{G} - 1_{\mathbb{G}}$ indicates the set of generators of $\mathbb{G}$.

5.5.3.2 Key derivation

On inputs msk and user's identity $id \in \{0,1\}^*$ (say IP address, etc.), the algorithm computes and returns the corresponding user's secret key sk_{id}.

$$sk_{id} = H_1(id)^\alpha \tag{5.5.1}$$

5.5.3.3 Signing

On inputs his/her own secret key sk_{id_i}, a message m, and the corresponding signature σ from an intended path $L = (id_1, ..., id_{i-1})$, the algorithm first parses σ as a three pairing computations (X, Y, Z)[6]. Then, the signer with identity id_i continues to performs the following steps:

- Pre-computation[7]:

$$s^{(i)} = s_1 s_2 \cdots s_i \tag{5.5.2}$$

  Where $s_j = H_2(id_1||id_2|| \cdots ||id_j)$ for $j = 1, 2, \cdots, i$.

- Choosing random number by the i^{th} signer: $r^{(i)} \in \mathbb{Z}_p^*$.

[6] For a first signer ($i = 1$), σ is defined as $(1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})$.

[7] If the intended signing order fixed, then s needs be computed only once; otherwise, whenever the intended signing sequence change, this pre-computation step needs to be re-executed.

- Computation:

$$X' \leftarrow H_1(m)^{r^{(i)} \cdot s^{(i)}} \cdot sk_{id_i} \tag{5.5.3}$$

$$Y' \leftarrow H_1(H_1(m))^{r^{(i)}} \cdot sk_{id_i} \tag{5.5.4}$$

- Finally, the algorithm returns

$$(X \cdot X',\ Y^{1/s_i} \cdot Y', Z^{1/s_i} \cdot g^{r^{(i)}}) \tag{5.5.5}$$

  Where $1/s_i$ means $s_i^{-1} \bmod p$.

Remark 5.5.3. To avoid potential forgery attacks, the online randomness r and the product $s = s_1 ... s_i$ are simultaneously involved in the computation X'. As a matter of fact, the use of r is necessary to protect or hide the information associated with the signing keys. While, the use of s helps to preserve the consistency and dependability of SFC (*e.g.*, a group of signer's identity on the intended path and the ordering in which they signed).

Remark 5.5.4. To make signature construction suitable for use in SFC domain, we consider multiple signers (*e.g.*, VNF appliances) sign each message/packet individually. Unlike the typically identity-based sequential aggregate signature which consider the situation that multiple signers can sign different messages simultaneously. Thus, the product s is mainly generated based on a hash function of a group of signers in accordance with their ordering (positions) on the intended path (Equation 5.5.2).

Iterative procedures in signing. In the following, we briefly explain step-by-step of generating aggregate signature when there are n-signers (*i.e.*, VNFs) involved in the service chain. For example,

- When the 1^{st} signer receives message or packet, he/she performs the signing process. To do that, let's initially define $\sigma = (X, Y, Z) = (1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})$ for the first signer, so that the first aggregate signature is generated as

$$\begin{aligned} X =& X \cdot X' \\ =& 1_{\mathbb{G}} \cdot H_1(m)^{r^{(1)} \cdot s^{(1)}} \cdot sk_{id_1} \\ =& H_1(m)^{r^{(1)} \cdot s^{(1)}} \cdot sk_{id_1} \\ Y =& H_1(H_1(m))^{r^{(1)}} \cdot sk_{id_1} \\ Z =& g^{r(1)} \end{aligned} \tag{5.5.6}$$

  Please note that since $1_{\mathbb{G}}$ is the identity element of $\mathbb{G}$, so that any multiplication of its identity elements do not affect the original value. That's say $X = 1_{\mathbb{G}} \cdot X' = X'$, and the processing of Y and Z are performed similarly.

- The 2^{nd} signer performs the same signing process using his/her secret key sk_{id} and the aggregate signature of the first signer, so that the newly aggregate signature is

$$
\begin{aligned}
X =& X \cdot X' \\
=& H_1(m)^{r^{(1)} \cdot s^{(1)}} \cdot sk_{id_1} \cdot H_1(m)^{r^{(2)} \cdot s^{(2)}} \cdot sk_{id_2} \\
=& H_1(m)^{r^{(1)} \cdot s^{(1)} + r^{(2)} \cdot s^{(2)}} \cdot sk_{id_1} \cdot sk_{id_2} \\
Y =& Y^{\frac{1}{s_2}} \cdot Y' \\
=& H_1(H_1(m))^{r^{(1)} \frac{1}{s_2}} \cdot sk_{id_1}^{\frac{1}{s_2}} \cdot H_1(H_1(m))^{r^{(2)}} \cdot sk_{id_2} \\
=& H_1(H_1(m))^{r^{(1)} \frac{1}{s_2} + r^{(2)}} \cdot sk_{id_1}^{\frac{1}{s_2}} \cdot sk_{id_2} \\
Z =& Z^{\frac{1}{s_2}} \cdot g^{r(2)} \\
=& g^{r^{(1)} \frac{1}{s_2} + r^{(2)}}
\end{aligned}
\tag{5.5.7}
$$

- Similarly, the aggregate signature of the 3^{rd} signer is generated as

$$
\begin{aligned}
X =& X \cdot X' \\
=& H_1(m)^{r^{(1)} \cdot s^{(1)} + r^{(2)} \cdot s^{(2)}} \cdot sk_{id_1} \cdot sk_{id_2} \cdot H_1(m)^{r^{(3)} \cdot s^{(3)}} \cdot sk_{id_3} \\
=& H_1(m)^{r^{(1)} \cdot s^{(1)} + r^{(2)} \cdot s^{(2)} + r^{(3)} \cdot s^{(3)}} \cdot sk_{id_1} \cdot sk_{id_2} \cdot sk_{id_3} \\
Y =& Y^{\frac{1}{s_3}} \cdot Y' \\
=& H_1(H_1(m))^{(r^{(1)} \frac{1}{s_2}) + \frac{1}{s_3}} \cdot sk_{id_1}^{\frac{1}{s_2 \cdot 2_3}} \cdot sk_{id_2}^{\frac{1}{s_3}} \cdot H_1(H_1(m))^{r^{(3)}} \cdot sk_{id_3} \\
=& H_1(H_1(m))^{(r^{(1)} \frac{1}{s_2} + r^{(2)}) \frac{1}{s_3} + r^{(3)}} \cdot sk_{id_1}^{\frac{1}{s_2 \cdot s_3}} \cdot sk_{id_2}^{\frac{1}{s_3}} \cdot sk_{id_3} \\
Z =& Z^{\frac{1}{s_2}} \cdot g^{r(2)} \\
=& g^{(r^{(1)} \frac{1}{s_2} + r^{(2)}) \frac{1}{s_3} + r^{(3)}}
\end{aligned}
\tag{5.5.8}
$$

- Therefore, the aggregate signature of the n^{th} signer can be generated based on the following forms

$$
\begin{aligned}
X =& H_1(m)^{r^{(1)} \cdot s^{(1)} + r^{(2)} \cdot s^{(2)} + \cdots + r^{(n)} \cdot s^{(n)}} \cdot sk_{id_1} \cdot sk_{id_2} \cdots sk_{id_n} \\
=& H_1(m)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}} \prod_{i=1}^{n} sk_{id_i} \\
Y =& H_1(H_1(m))^{(\cdots((r^{(1)} \frac{1}{s_2} + r^{(2)}) \frac{1}{s_3} + r^{(3)}) \frac{1}{s_4} + \cdots) \frac{1}{s_n} + r^{(n)}} \cdot sk_{id_1}^{\frac{1}{s_2 \cdot s_3 \cdots s_n}} \cdots sk_{id_{n-1}}^{\frac{1}{s_n}} \cdot sk_{id_n} \\
& \textbf{Let } w \triangleq (\cdots((r^{(1)} \frac{1}{s_2} + r^{(2)}) \frac{1}{s_3} + r^{(3)}) \frac{1}{s_4} + \cdots) \frac{1}{s_n} + r^{(n)} \\
=& H_1(H_1(m))^{w} \prod_{i=1}^{n} sk_{id_i}^{\frac{1}{\prod_{j=i+1}^{n} s_j}} \\
Z =& g^{(\cdots((r^{(1)} \frac{1}{s_2} + r^{(2)}) \frac{1}{s_3} + r^{(3)}) + \frac{1}{s_4} + \cdots) \frac{1}{s_n} + r^{(n)}} \\
=& g^{w}
\end{aligned}
\tag{5.5.9}
$$

5.5.3.4 Verification

On inputs mpk, a message m, and the corresponding signature σ from the intended path $L = (id_1, .., id_n)$, the algorithm first returns 0 if all of $id_1, ..., id_n$ are not distinct. This check is needed to ensure that there are no signers repetition occurred during an aggregate signature construction. As a matter of fact, this event can constitute a security vulnerability and should not be allowed anyway. If the above condition is met then it parses σ as (X, Y, Z), and a verifier continues to proceed the following steps:

- Pre-computation[8]:

$$S = \prod_{i=1}^{n} H_1(id_i)^{\frac{1}{\prod_{j=i+1}^{n} s_j}} \tag{5.5.10}$$

$$T = \prod_{i=1}^{n} H_1(id_i) \tag{5.5.11}$$

$$s = s^{(n)} = s_1 s_2 \cdots s_n \tag{5.5.12}$$

  where $s_j = H_2(id_1||id_2|| \cdots ||id_j)$.

- Verification: the algorithm checks whether the following two equations hold true simultaneously.

$$e(Y, g) \stackrel{?}{=} e(H_1(H_1(m)), Z) \cdot e(S, g^{\alpha}) \tag{5.5.13}$$

$$e(X, g) \stackrel{?}{=} e(H_1(m), Z^s) \cdot e(T, g^{\alpha}) \tag{5.5.14}$$

  If not, the algorithm returns 0. Otherwise, it returns 1 indicating that an aggregate signature σ is valid on the message m with respect to the intended path, authenticated signers and their ordering properties $id_1, \cdots, id_n$.

  Remark 5.5.5. As we discussed earlier, in order to accelerate the construction time, our scheme does not verify the validity of aggregate signature during its signing algorithm. The verification process only performs once by the verifier.

5.5.4 Theoretic proofs

In this Section, we provide a mathematical proof to illustrate how our proposed scheme of Lite identity-based ordered multisignature can help to fully preserve both security (*e.g.*, authenticity, integrity of VNF), dependability (*e.g.*, ordering property) and consistency (*e.g.*, VNF chaining/composition) in service chain. Suppose that the aggregate signature of the n-th signer is $\sigma = (X, Y, Z)$ in which

$$X = H_1(m)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}} \prod_{i=1}^{n} sk_{id_i} \tag{5.5.15}$$

$$Y = H_1(H_1(m))^{w} \prod_{i=1}^{n} sk_{id_i}^{\frac{1}{\prod_{j=i+1}^{n} s_j}} \tag{5.5.16}$$

[8] If the intended signing order fixed, then S, T and s needs be computed only once; otherwise, whenever the intended signing sequence change, this pre-computation step needs to be re-executed.

$$Z = g^w \tag{5.5.17}$$

Where w is defined by an Equation 5.5.18.

$$w \triangleq (\cdots((r^{(1)}\frac{1}{s_2} + r^{(2)})\frac{1}{s_3} + r^{(3)})\frac{1}{s_4} + \cdots + r^{(n-1)})\frac{1}{s_n} + r^{(n)} \tag{5.5.18}$$

Now, we need to prove that whether the Equation 5.5.13 and Equation 5.5.14 hold true simultaneously. Let's first calculate the left-hand side of an Equation 5.5.13 using substitution method derived from Equation 5.5.16, finally we get Equation 5.5.19.

$$\begin{aligned} e(Y, g) =& e(H_1(H_1(m))^w \prod_{i=1}^{n} sk_{id_i}^{\frac{1}{\prod_{j=i+1}^{n} s_j}}, g) \\ =& e(H_1(H_1(m))^w, g) \cdot e(\prod_{i=1}^{n} sk_{id_i}^{\frac{1}{\prod_{j=i+1}^{n} s_j}}, g) \\ =& e(H_1(H_1(m)), g)^w \cdot e(\prod_{i=1}^{n} H_1(id_i)^{\frac{1}{\prod_{j=i+1}^{n} s_j}}, g)^\alpha \end{aligned} \tag{5.5.19}$$

Similarly, on the right-hand side of the Equation 5.5.13, we substitute it using Equation 5.5.17 and Equation 5.5.10 respectively. Finally, we get the Equation 5.5.20.

$$\begin{aligned} e(H_1(H_1(m)), Z) \cdot e(S, g^\alpha) =& e(H_1(H_1(m)), g^w) \cdot e(\prod_{i=1}^{n} H_1(id_i)^{\frac{1}{\prod_{j=i+1}^{n} s_j}}, g^\alpha) \\ =& e(H_1(H_1(m)), g)^w \cdot e(\prod_{i=1}^{n} H_1(id_i)^{\frac{1}{\prod_{j=i+1}^{n} s_j}}, g)^\alpha \end{aligned} \tag{5.5.20}$$

Based on the results obtained from Equation 5.5.19 and Equation 5.5.20, we can conclude the received aggregate signature holds true for Equation 5.5.13.

Furthermore, according to Equation 5.5.14, we perform the same way as described above, to further calculate the results on both sides of the equals sign. Thus, using substitution method of X (Equation 5.5.15), Z (Equation 5.5.17) and T (Equation 5.5.11), we finally obtain Equation 5.5.21 and Equation 5.5.22 respectively.

$$\begin{aligned} e(X, g) =& e(H_1(m)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}} \prod_{i=1}^{n} sk_{id_i}, g) \\ =& e(H_1(m)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}}, g) \cdot e(\prod_{i=1}^{n} sk_{id_i}, g) \\ =& e(H_1(m), g)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}} \cdot e(\prod_{i=1}^{n} H_1(id_i), g)^\alpha \end{aligned} \tag{5.5.21}$$

$$\begin{aligned} e(H_1(m), Z^s) \cdot e(T, g^\alpha) =& e(H_1(m), g^{w \cdot s}) \cdot e(\prod_{i=1}^{n} H_1(id_i), g^\alpha) \\ =& e(H_1(m), g)^{\sum_{i=1}^{n} r^{(i)} s^{(i)}} \cdot e(\prod_{i=1}^{n} H_1(id_i), g)^\alpha \end{aligned} \tag{5.5.22}$$

Where w can be derived from the Equation 5.5.18, and we found that $w \cdot s$ is equivalent to $\sum_{i=1}^{n} r^{(i)} \cdot s^{(i)}$ with regards to Equation 5.5.23.

$$\begin{aligned} w \cdot s =& ((\cdots((r^{(1)}\frac{1}{s_2} + r^{(2)})\frac{1}{s_3} + r^{(3)})\frac{1}{s_4} + \cdots + r^{(n-1)})\frac{1}{s_n} + r^{(n)}) \cdot s \\ =& (\cdots((r^{(1)}\frac{1}{s_2} + r^{(2)})\frac{1}{s_3} + r^{(3)})\frac{1}{s_4} + \cdots + r^{(n-1)})\frac{1}{s_n} \cdot s + r^{(n)} \cdot s \\ =& (\cdots((r^{(1)}\frac{1}{s_2} + r^{(2)})\frac{1}{s_3} + r^{(3)})\frac{1}{s_4} + \cdots + r^{(n-1)})\frac{1}{s_n} \cdot s^{(n)} + r^{(n)} \cdot s^{(n)} \\ =& (\cdots((r^{(1)}\frac{1}{s_2} + r^{(2)})\frac{1}{s_3} + r^{(3)})\frac{1}{s_4} + \cdots + r^{(n-1)}) \cdot s^{(n-1)} + r^{(n)} \cdot s^{(n)} \\ =& \sum_{i=1}^{n} r^{(i)} \cdot s^{(i)} \end{aligned} \tag{5.5.23}$$

Observing the results obtained from both Equations (Equation 5.5.21 and Equation 5.5.22), we can conclude that the received aggregate signature holds true for Equation 5.5.14. Thus, the security, dependability and consistency can be preserved if the two above bilinearity conditions of pairing are satisfied.

To summarize, Lite identity-based ordered multisignature is proposed which aims at examining the behavior of packet traversal. Let's say whether the messages/packets associated to a particular service chain have indeed traversed through all intended VNF appliances as specified in the SFC policy. Thanks to the verifiability properties of signing order and ordinary signature which provide a verifier the ability to examine the consistency of service chain and to ensure that only the *honest* participants are able to produce a valid aggregate signature on that message. Since the verifier knows a certain number of signer's identity (say a set of IP addresses of VNF appliances involved in a particular service chain) and their sequences, so that it can straightforwardly verify its consistency from the bilinearity condition of a pairing. As a result, any missing, detouring, bypassing one or more VNF nodes along the intended path, out-of-order traversal, or the occurrence of impersonating the target VNF nodes can lead to unsatisfied condition under bilinear maps.

5.6 Security Analysis

In this section, we analyze the security and justify how our proposed scheme of Lite identity-based ordered multisignature can be used to preserve security, dependability and consistency in SFC, further protecting against security threats described in Section 5.4.2.

On the length of keys and aggregate signature. The proposed scheme is considered as an extension of two combination schemes (ordered multisignatures and identity-based sequential aggregate signatures), so that the key features of constant-sizes in signature and keys having in the two above schemes are also inherited to our proposal. However, it appears that the proposed scheme is substantially more efficient, providing lightweight and cost-effective signature construction when compared to all existing aggregate signature alternatives including the two above-mentioned schemes [22]. It yields constant-size aggregate signature of three group elements. For example, it requires about $|msk| = \log p$, $|sk_{id}| = \log p$, and $|\sigma| = 3 \log p$, for the sizes of master public key msk, user's secret key sk_{id}, and the aggregate signature σ, respectively.

Bypass elements. Suppose that a packet Pkt is received by a compromised switch which then forwards Pkt directly to VNF_{i+1}, thereby bypassing the original next hop VNF_i. According to the proposed scheme and its verifiable properties, if one of the signers (*i.e.*, VNF

nodes) involved in the service chain have been bypassed, it can be easily detected by the verifier. Recall the signature construction, to compute an aggregate signature the product s needs to be involved, where s implicitly related to the signers' identities and their ordering properties. The signing processes are iteratively repeated by all authenticated members along the intended service function path. At the end, the verifier can examine the behavior of packet traversal whether Pkt is indeed traversed through all authenticated VNFs according to the specified SFC policy, thereby performing a similar operation of reconstructing the signature and further comparing the result with the received one using bilinearity condition mapping (Equation 5.5.13 and Equation 5.5.14). Thus, the occurrence of bypassing can ultimately lead to unsatisfied conditions in a bilinear map.

Path detour. Similar to the above case, but little difference in which the packet has been bypassed the original intended VNFs, and unlawfully performing by anonymous service functions which are not actually specified in the SFC policy. Suppose a packet Pkt is intended to traversed along the service function path as $SFC_i : \{VNF_1 \rightarrow \cdots \rightarrow VNF_i \rightarrow \cdots \rightarrow VNF_n\}$. However, when a compromised switch received Pkt, it deviates Pkt to the anonymous VNF_i'' node instead of the original service function VNF_i. In this case, the aggregate signature σ' would be different from the σ generated by the original path and will not pass the verification with high probability. Thanks to its verifiable property, path detour can be detected. Recall the signature construction, a newly aggregate signature of the current hop is actually based on the result of aggregate signature from the previous hops. This ensures that only the honest participants are able to reconstruct the valid aggregate signature. Although Pkt has been deviated to anonymous VNF_i'' node, it does not possess the key share (msk) which is considered as essential input for further generating the corresponding user's secret key sk_{id} and producing an aggregate signature σ. As a result, it cannot generate a valid σ. In other words, even Pkt has been detoured, it can still be detected by the verifier using bilinear maps.

Out-of-order traversal. Regarding the verifiable properties of Lite identity-based ordered multisignature, even the out-of-order traversal occurs, this kind of attack can easily be detected by the verifier. As aforementioned, hence the verifier knows well about the list of VNFs and their positions on the intended path, so that it can reconstruct sequential aggregate signature using the corresponding master public key share mpk (the right-hand sides of Equation 5.5.13 and Equation 5.5.14) and comparing the result with the received value (the left-hand sides of Equation 5.5.13 and Equation 5.5.14). In case of out-of-order traversal, the two values would be different, leading to failed verification with 0 bit returned. Even the same VNF node and the same packet, if its position has been modified, the final result of aggregate signature is totally different from what would be generated when Pkt traversed through the original path in the correct order.

Packet replay. In which a passive attacker record previous packet flows and replays them into the network. The objective here is to let these replayed flows successfully pass the verifier's check. To do that, he may act as a man-in-the-middle to intercept the intermediate aggregate signature $\widehat{\sigma}$ and reusing them for future verification. However, according to our proposed scheme, it is impractical to pass this verification with high probability. Despite he can intercept $\widehat{\sigma}$ from the previous hops, the way of reconstructing aggregate signature at the next-hop downstream VNFs are not only based on $\widehat{\sigma}$, but also the master secret key msk and user's secret key sk_{id} as well. Unfortunately, the passive attacker does not know these keys, so that he cannot produce a valid σ. As a result, the replayed packet can be detected.

5.7 Implementation and Evaluation

In this Section, we present the proof of concept (PoC) implementation of our proposal over NFV/SDN environment, and evaluate the performance of our proposed scheme in terms of end-to end latency.

5.7.1 Implementation details

The correctness of our digital signature scheme can be theoretically proved, but its implementation in real environment is non-trivial, and we need to solve the following technical challenges, (1) generate and distribute the cryptography parameters to each involved VNF node, (2) add signature generation into VNF nodes and, (3) transmit packets with signature. To address these challenges, we propose the following solutions.

A trusted Private Key Generator (PKG). The first challenge requires a PKG to generate its master public key mpk and the corresponding master secret key msk. Such a PKG server is developed and run alongside the ODL SFC controller. To create a service chain, a SFC description file including information identities of each service function, the SPI associated with a particular service chain and SI which is used to determine the next VNF to be traversed, is simultaneously sent to PKG and ODL controller. With these information, PKG generates cryptography parameter settings (elliptic curve type, key size, user's secret key sk_{id}, *etc.*), and distribute them to each involved VNF through REST API. The implementation of PKG is built on top of PBC (Pairing-Based Cryptography) library [138], PyPBC (Python binding for PCB) [185] and Python Flask web framework.

NSH and OMS aware VNF. In our implementation, the first VNF on a given chain is in charge of generating and inserting the aggregate signature. Then the next hop downstream VNFs should be able to retrieve the signature from the received packet and update it with the new one generated based on the construction method described in Section 5.5.3. Thus, we developed some NSH and OMS (Ordered Multisignature) aware VNF. An instance of this kind VNF is able to parse the NSH header and choose the appropriate actions, *e.g.*, generating, inserting or updating the aggregate signature. The signature verification is done at the last classifier on a chain, namely just before leaving SFC domain. In our implementation, a new verification service is developed as an extended function and deployed alongside of the egress classifier to take charge of signature validation. In the presence of attacks, an alert message w.r.t the failed signature validation should be displayed. In addition, a message m which is used as input in signing process can be any given value if it is known by all the VNF nodes on a particular chain. The NSH payload is not suggested to be used as the input of signing process, because the content of playload may be modified by some VNFs (for example, NAT) along service path. This can lead to the failure of signature verification.

Extended NSH type 1 metadata. To use our proposed scheme, the signature should be inserted into the packets. In the current implementation, we leverage the metadata field of NSH header (as shown in the right below corner of Fig. 5.5), which has two types according to the specification of RFC8300, (1) NSH MD type-1, which has fixed 16 bytes length; (2) NSH MD type-2, which has a variable length. However, Type-1 has no sufficient room to occupy the signature, while Type-2 has not been implemented so far. Therefore, we extend NSH MD type-1 with an extra variable signature field. Also, we use one non-reserved bit in the NSH base header as *signature bit*, indicating whether the packet carries the signature (value 1) or not (value 0). If a packet with signature is received by a VNF, the signature will be extracted from the NSH header and updated with the newly generated one using the VNF's secret key sk_{id} shared by the PKG at the SFC initialization stage.

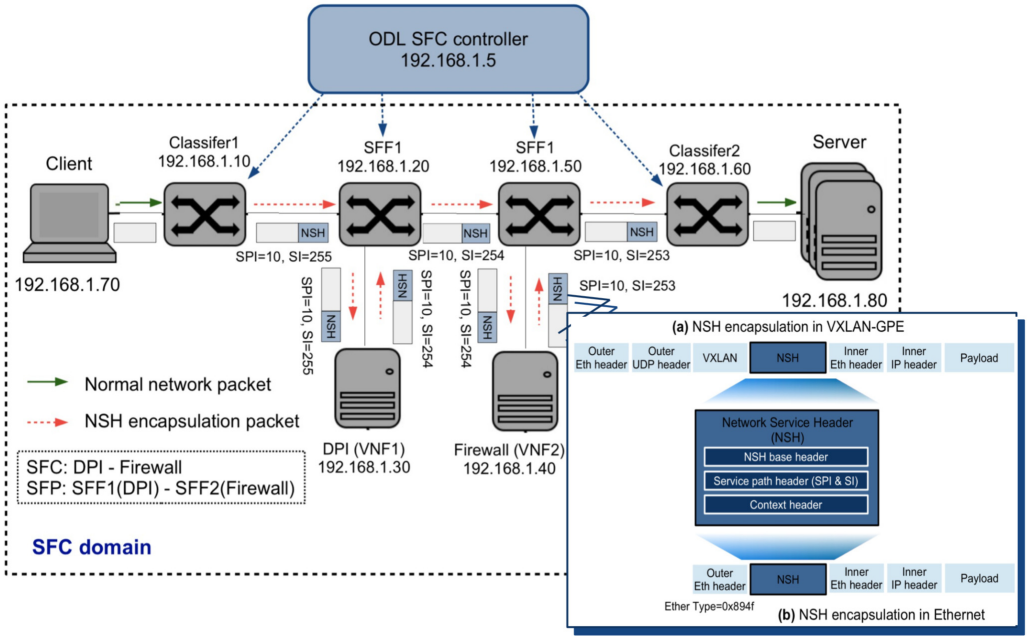


Figure 5.5: Deployment scenario

5.7.2 Performance evaluation

To present our prototype implementation, we build a testbed as shown in Fig 5.5. It is developed based on the VirtualBox virtualization environment [175] and the Vagrant tool [101]. The hardware specification of the host system that runs the testbed is based on Linux desktop with 2.5GHz Intel Core i7 CPU and 16G RAM. All the deployed network nodes including controller, VNFs, classifiers and SFFs[9], are implemented as docker containers running inside Vagrant VM with Ubuntu/xenial 64, each node has been customized with 2 CPU cores and 4G RAM. Our motivation to use OpenDaylight (version Fluorine) [226] as a ODL SFC controller is that it is developed with fully NSH encapsulation support, which allows us to use it for developing a prototype that carries an aggregated signature generated from our proposed approach.

5.7.2.1 Computational overhead.

To evaluate the performance of our proposed scheme, we ran a set of experiments and examined the relationship between the processing capacity of our proposed scheme and the number of VNF nodes involved in a service chain. In practice, we used PBC library with Type A pairings to create a group of $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ on the rational points of a elliptic curve $y^2 = x^3 + x$ over a finite filed. Using embedding degree (the degree of certain extension of the ground filed), which is $k = 2$, thus the elements in $\mathbb{G}$ can be respectively represented using about 512, 256, and 160 bits to achieve the equivalent of standard security level, 256, 128, and 80 bits security [13]. Please note that by reducing the bit-length of elements in $\mathbb{G}$ the pairing computation complexity can be reduced and of cause further reducing the security level as well. However, to meet the lower bound of security strength, a minimum of 80 bits security are required, equivalent to the size of elliptic curve keys of 160 bits.

We ran the experiment 50 epochs with different key sizes[10] and calculate the 95% confidence interval. As shown in Fig. 5.6, we observed that the lower bound latency goes linearly. It strongly depends on three major factors: (1) the employed cryptography parameter settings

[9] Classifiers and SFFs are used to install Open vSwitch (OVS). Please note that NSH – service chaining encapsulation protocol for SFC, is supported in OVS 2.9 and higher.

[10] Both elements in $\mathbb{G}$ and keys contains the same bit length.

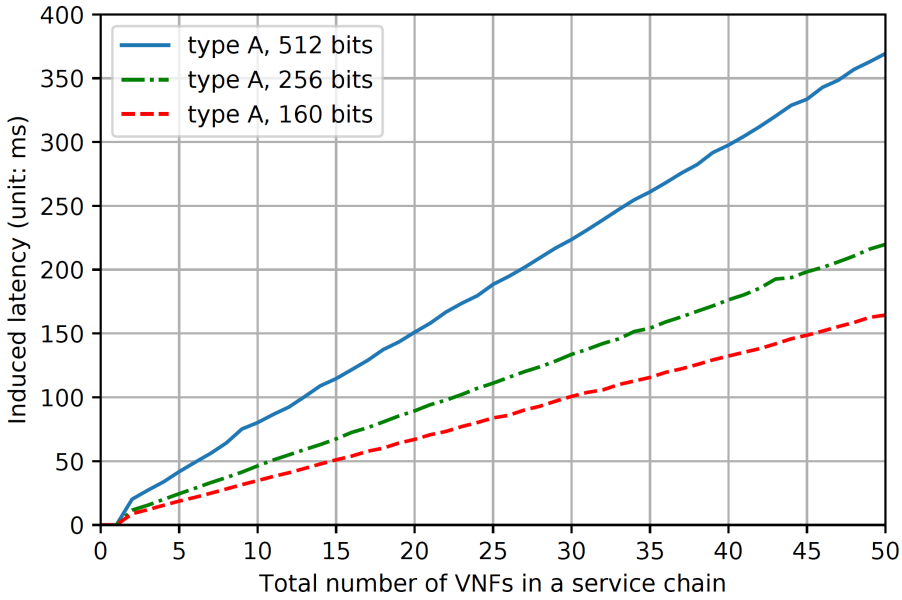


Figure 5.6: The relationships between number of VNF nodes in a service chain and latency

which include the type of pairing operations over the specified elliptic curve, the length of elements in $\mathbb{G}$ and keys, as well as signing and verification algorithms; (2) the total number of VNF nodes involved in a service chain; and (3) the processing delays caused by signing operations conducted by each VNF node and also the verification operation at the last egress classifier. For example, if we only consider the processing delay related to signature and verification constructions, without taking into account other parameters such as the parameter settings, times for packet manipulation (*e.g.*, inserting, parsing or updating the aggregate signature into NSH header) and transmission, the overall processing delays for a particular service chain $\mathcal{D}_{SFC_i}$ can be obtained in the following form,

$$\mathcal{D}_{SFC_i} = \sum_{i=1}^{N} d_{s_i} + d_v$$

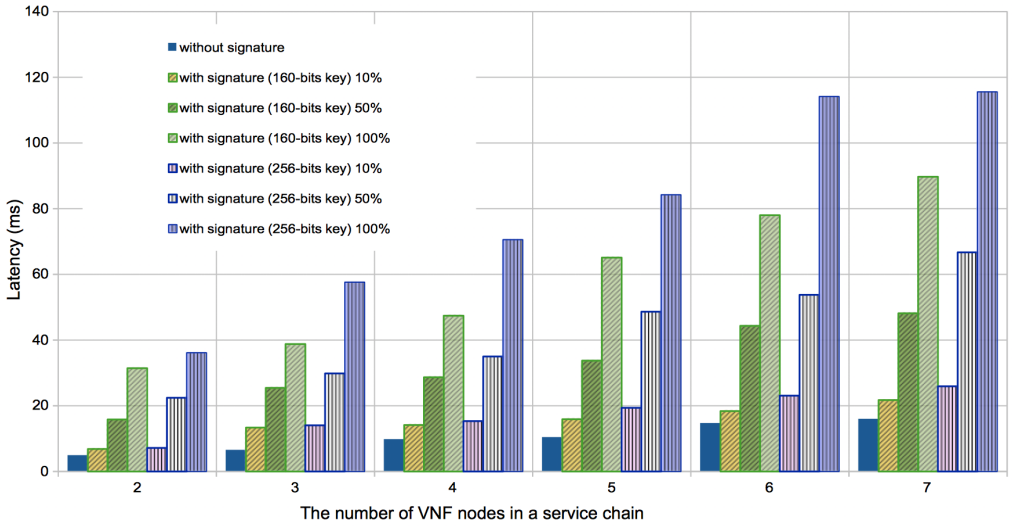
where d_{s_i} is the delay time takes by each VNF node to complete signing operation, and d_v is the delay time to perform verification at the last verifier node, *e.g.*, validating the aggregate signature.

In practice, the number of VNFs nodes involved in a service chain is normally less than 10 (the exemplified SFC use cases are given in Section 5.3 and in [95, 127, 233]). For example, to achieve the lowest bound of security level (80 bits), it took around 34.76 ms to successfully validate an aggregated signature when 10 VNF nodes were involved in the service chain. This latency is minimal and acceptable for most of network services.

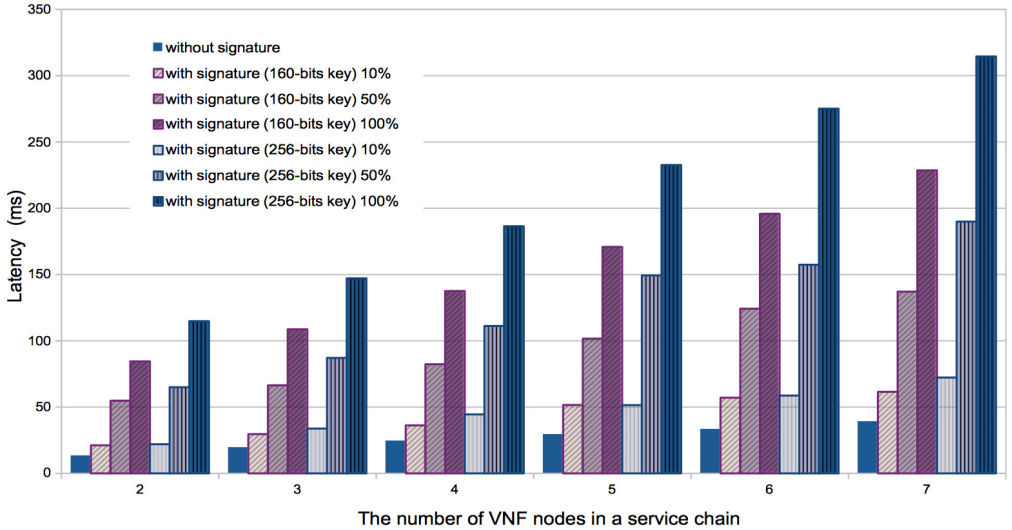
5.7.2.2 Latency.

To evaluate the end-to-end packet transmission latency brought by the proposed scheme. We run 6 groups of experiments, in which the number of VNF nodes involved in a service chain is varied from 2 to 7. Initially, we used network topology illustrated in Fig. 5.5 as a starting point in order to create a simple service chain of connecting 2 VNF nodes, and consequently increased the number of VNFs to 7. Not that these numbers of VNFs in a service chain are reasonable value when considering several typical SFC deployment use cases discussed in [136].

Within each group experiment, the case without signature scheme is used as comparison reference. We consider two OMS enabled cases with different elliptic curve key sizes (*i.e.*, 160 bits and 256 bits). It should also be noted that the proposed scheme enforces all the involved VNFs to sign the received packets. However, there is no need to let the instantiated VNFs always sign every processed packets. Recall our main objective of the proposed scheme, which aims to provide secure, consistent and dependable SFC against anomalous flow redirection and path deviation (as discussed in Section 5.4.2). With the presence of attack, the packets with signatures are detected due to the failed signature validation. As a result, there is no need to sign and verify every packets over the given service function path. In other words, one of the main reason of discussing such this aspect is because we need to reduce as much as possible the end-to-end latency related to the service chain, we then apply a probabilistic method into our scheme. At the initialization stage, the first VNF is configured to sign a received packet based on a given probability. That says, in our experiment, we consider the cases of signing probability 10%, 50% and 100% respectively.



(a) Transmission of ICMP-ping packets



(b) Transmission of HTTP-ping packets

Figure 5.7: Two different types of packet transmission (*i.e.*, ICMP-ping and HTTP-ping packets) with and without signature construction. Percent such as 10% means that for each received packet, first VNF signs it with probability 10%.

To evaluate the performance, we launch two different tests based on two types of transmission protocols: (1) ICMP-ping test, and (2) HTTP-ping test. For each test, we continuously sent 100 ping messages from client to server and measured the end-to-end packet transmission latency. The results are illustrated in Fig. 5.7a and Fig. 5.7b, respectively. To achieve the lower bound of security level (160 bits elliptic key), if signing probability is set as 100%, 50%, and 10% respectively[11], on average the latency of ICMP-ping respectively increased around 5.80, 3.25, and 1.52 times when compared to a conventional packet transmission without signature. Similarity, it respectively increased about 5.89, 3.63, and 1.62 times for HTTP-ping. The experimental results showed that the overall latency has been significantly reduced, especially with 10% signing probability, the resulting latency can be almost ignored. In addition, the average latencies between two types of packet transmission were not much different. The probability of signing operation depends on its setting (*e.g.*, 10%). Thus, no matter the type of packet transmission is, the signing process is randomly performed based on the total number of packets. For example, if 10% signing probability is set, then on average only 10 out of 100 packets will be signed.

The setting of 10% or 50% signing probability is our preliminary exploitation about how to efficiently integrate our proposed scheme into the real world SFC deployment. However, the setting of signing probability value depends on type of SFC use cases. For example, if it is video-based application use case, this value should be set as small as 0.1%. In case of smart metering use case that may upload a message per hour even per day, this value can be set as 100%. In the future work, we plan to improve our proposed scheme with a probabilistic model in order to find the optimal tradeoff between the transmission overhead and detection performance.

5.8 Conclusion and Open Issues

In this Chapter, we proposed a solution to solve a specific problem in Service Function Chaining, which means a secure, consistent and dependable SFC. In ideal case, a number of VNFs (belonging to different service providers in different administrative domains) can be chained together for implementing a particular network service. One of the key challenges is to ensure that only the authentic VNFs are put together in a correct and optimal order. In another word, the packet flows sequencing, associated with a particular service should be preserved, according to the pre-specified SFC policy. However, an attacker can redirect the malicious flows and change the defined path in order to evade some security functions, *e.g.*, firewall, DPI, IDS, or for any other malicious purpose. We therefore presented and described a scheme based on *Lite identity-based ordered multisignature* cryptographic primitive. That says, all the VNFs in the service chain need to generate and attach the signature into the packets they receive. The signature is kept with a compact and constant size, and can be inserted into NSH header. The last hop NFV serves as a verifier verifying the signature, while any anomalous flow redirection or path deviation can lead to unsatisfied conditions and ultimately failed verification. The scheme is essentially distributed, bringing no burden to PKI and key management. More importantly, as the computational overhead w.r.t signature generation and verification is minimal, the resulting latency is limited for most of typical network services.

It is worth noting, however, NSH is still under development and standardization, so the implementation aspects of our scheme need to be developed and improved. With the support of NSH, developing different use cases for deploying and testing our scheme will be next research objective. In addition, optimal SFC is an active, important and interesting research topic. How

[11] Note that, if the signing probability is set as 100%, means every packets have to be signed, while 10% and 50% indicate that on average only 10 and 50 out of every 100 packets will be signed.

to integrate our scheme with optimal SFC chaining algorithms is still an open issue that we will study in future works.

6 Conclusion and Future Work

This Chapter summarizes the major research contributions of this thesis and further discusses about future research directions.

6.1 Research Contributions

The emergence of NFV and SDN yields numerous benefits. One of the most salient features is the cost-efficient transition from dedicated hardware appliances to software based approach, which greatly reduces the total cost of ownership by having the potential to break vendor lock-in, while improving service agility and scalability to meet dynamic changes in business requirements. As a two-sided coin, despite these well-recognized benefits, security remains to be one of the vital concerns and potential impediments in development and deployment of network services over the NFV environment. Therefore, this thesis is dedicated to exploring the pros and cons of NFV/SDN from security perspective. On the one hand, we explored the potential security issues in NFV, established a cross-layer threat taxonomy, along with a set of security hardening recommendations resulting from a gap analysis between the desirable security requirements and available security mechanisms. On the other hand, we developed a conceptual framework SecMANO, as a natural extension of TOSCA based NFV MANO frameworks, to manage and orchestrate those security functions. Based on SecMANO, a security orchestrator has been further developed and implemented, having a novel access control paradigm at its core. In addition, a security scheme based on ordered aggregated signature was developed to ensure secure and dependable Service Function Chaining (SFC). Generally, the contributions can be organized in the following way,

- **Security in NFV.** Considering the fact that NFV and SDN have drawn significant attention from both industry and academia as next generation network architecture that facilitate the deployment of numerous network applications with high flexibility, scalability, manageability, and agility, it is urgent and important to identify the potential security threats and vulnerabilities that will be introduced. It is also important to have a clear understanding about the NFV attack surface and to what extent the available NFV security best practices can fulfil the security recommendations. To achieve the objectives, we have investigated all the nine use cases specified by ETSI and deeply studied five of them for identifying their vulnerabilities. Then a NFV layer specific threat taxonomy was established, together with the corresponding security recommendations. Meanwhile, the state-of-the-art security mechanisms have been analyzed in terms of their potential usability in NFV environments. We believe this is one of the first contributions in this domain, and it mainly serves for three primary purposes: (1) to help service providers and network operators

to gain a holistic understanding on the attack surface of NFV; (2) to allow them to deploy cost-effective security hardening according to their particular business needs; and (3) to develop novel security countermeasures tailored to NFV services and integrate them together for achieving multi-layered, optimal, flexible, and adaptive defense.

- **NFV based Security Management and Orchestration.** We assume that NFV and SDN will gain popularity in the next few years and have potent to significantly reshape today's ICT infrastructure. In the new networking infrastructure, security management must be reconsidered. As NFV MANO is the core part of NFV architecture, which plays an important role in orchestrating the underlying infrastructure resources, and maintaining the full lifecycle management of VNFs and network services to achieve on-demand service delivery, we started with the analysis and comparative studies on the existing NFV MANO frameworks in terms of architecture, data model, and functional components, and we found that none of the existing NFV orchestrator that can provide holistic and dedicated security management. We then leveraged the capabilities of NFV (*e.g.*, flexibility, scalability, adaptability) and SDN (*e.g.*, programmability, global visibility, centralized control) to develop a novel security management framework called SecMANO, which can be treated as a natural extension of the existing NFV orchestrators working with TOSCA model (one of the popular data models for cloud orchestrators). One of the expected roles of SecMANO is empower the NFV orchestrator to have the capability of security management, so that the most appropriate security functions can be dynamically orchestrated and deployed in NFV environment. We did not assume any specific threat models or attack models for SecMANO, because it's a general purpose design and intended to achieve two desirable properties, (1) *security by design* by formally specifying security attributes of interest a the early stage of service deployment, ensuring that all the deployed assets (*e.g.*, VMs, VNFs) are associated with certain security attributes; and (2) *security as a service* which aims at providing a set of well-defined security functions on demand to help protect the network resources and services based on particular security requirements and threat models.

- **Security orchestrator for achieving software-defined access control.** This part of work essentially contains two contributions: a TOSCA model compliant security orchestrator, and a novel access control paradigm. Specifically, to implement our conceptual framework SecMANO, we developed a security orchestrator that contains two major components: (1) a TOSCA-parser, which aims at extracting the security attributes of VM/VNF nodes from the given TOSCA files and parsing them to an access control engine to further generate the corresponding access control policies based on tenant-specific access control model, meanwhile the necessary information other than security attributes are input to NFV orchestrator; and (2) a novel software-defined tenant-specific access control paradigm, which allows security administrators to dynamically customize the access control models and policies for different tenant domains, ultimately achieving flexible and scalable protection across different NFV layers and multiple cloud data centers. In particular, the first component is now limited to TOSCA data models, while we expect to have such a component in future orchestrators. The second component, *i.e.*, a new access control paradigm, is adapted to our security orchestrator. Access control models and policies can be specified according to the particular needs of the customers. The security orchestrator was then prototyped with Tacker (an OpenStack service for NFV orchestration using TOSCA model), and its performance was evaluated in terms of three primary metrics, *i.e.*, throughput, scalability, and adaptability. The experiments demonstrated that the security orchestrator could maintain a satisfactory level performance regardless of the varying number of tenants, users and objects in the cloud.

- **Towards secure and dependable Service Function Chaining (SFC).** Our NFV cross-layer threat taxonomy indicates that a large set of novel threats can be potentially introduced to NFV environment, among which VNFs and their chaining properties (in other terms, it known as service function chaining or VNF forwarding graph) attract special interests and attentions. A natural question, for example, is that once the high-level SFC policy is specified, how can we make sure that the packet flows of a particular service chain are indeed traversed through all the specified VNFs in the right order. To violate such a policy, the attackers can manage to redirect the packet flows and lead to path deviation. To address the issue, we proposed a new security primitive, called *Lite identity-based ordered multisignature* scheme to provide secure, consistent and dependable SFC in NFV and SDN environment. The objective is to ensure that each service function involved in a particular service chain is authenticated and legitimate, and all service functions are chained in a consistent and reliable way. The design foundation of the proposed scheme aimed at enforcing a group of service functions involved in a service chain to attest their signatures on the packets received. At the end, a verifier can later verify whether the packets are traversed correctly with regard to the specified SFC policy. In addition, the proposed scheme was developed to achieve five key properties: (1) unforgeability, which is computationally infeasible for any adversary to produce a forgery; (2) authenticity of service functions involved in a particular service chain; (3) re-order protection, it is not possible to re-order the position of the legitimate signers, e.g., VNFs; (4) compact and constant-size keys and aggregate signatures; and (5) signature and verification acceleration using three pairing computations in bilinear maps. The experimental results show that the proposed scheme can achieve a satisfactory level of security with small computational overheads on both signature and verification regardless of the path length.

6.2 Perspectives

With the technological trend of software-defined everything, software-defined security has emerged as a new concept attracting increasing efforts from both industry and academia. Security controllers in SDN and security orchestrators in NFV can be both treated as stepping stones towards software-defined security. However, *Roma is not built in one day*. Much more efforts are desperately expected. As the subsequent work to the contributions reported in this thesis, the following topics have been identified and will be further studied.

- **Towards trustworthy SFC.** In Chapter 5, we proposed a new security primitive, called Lite identity-based ordered multisignature scheme to provide secure, consistent and dependable SFC. We implemented the proposed scheme in NFV and SDN environments using OpenDaylight Service Function Chaining (ODL SFC) that supports Network Service Header (NSH) protocol [188], which is used to carry the aggregate signature generated by those service functions involved in a particular service chain. It is worth nothing, however, the current ODL SFC framework is still under development, it provides limited feature and only supports a fixed length metadata with 16 bytes (NSH MD type 1). As such, the generated signature has larger size that the metadata field[1]. An alternative solution that we presented in the preliminary implementation is to extend NSH MD type 1 with an extra variable signature field and used one non-reserved bit in the NSH base header as signature bit. In the future, we expect to leverage NSH MD type 2 to implement our scheme and evaluate its performance in several different use cases, *e.g.*, web service, email, video

[1] According to NIST standard [13], in order to achieve the lower bound of security level, a minimum of 80 bits is required, leading to the size of ECC key 160 bits and around 480 bits in signature. Please note that based on the proposed scheme, the signature is combined from the three elements (X, Y, Z), each element occupied 160 bits.

service. We also expect to improve our proposal with an efficient probabilistic model to find the optimal tradeoff between the transmission overhead and detection performance.

- **Enriching the capability of our security orchestrator.** (1) In the current version security orchestrator, we have only evaluated the performance based on an access control model RBAC. We intend to further study and evaluate more access control models for providing greater flexibility protecting NFV assets in the distributed clouds. (2) In addition to access control, other security functions like security monitoring, IDS/IPS, network isolation, firewall, and data protection are expected to be implemented and deployed in our security orchestrator. For example, a security orchestrator can implement and deploy a number of different sniffers or monitoring functions to monitor the events related to the system behavior of physical/virtual appliances. Then the events can be filtered and aggregated for further forensic analysis and anomaly detection. Similarly, a set of diverse anomaly detectors can be deployed at different NFV layers (*e.g.*, infrastructure, VNF). The detection alerts can be optimally correlated for achieving broader detection coverage and lower false positive rate. As a result, if any anomalous event is detected, an appropriate security countermeasure will be automatically initiated, according to the pre-defined security policies, to defend against and mitigate it.

- **Development of interesting use cases for our security orchestrator.** In Chapter 4, we have reported the implementation and deployment of our security orchestrator in the distributed cloud data centers. Considering the fact that NFV and SDN have become driving forces for other emerging networking technologies such as 5G, it would be very interesting to deploy our security orchestrator in 5G related use cases. For example, network slices or micro segments are empowered by 5G, which share the same network infrastructure but provide different network services, *e.g.*, VANETs, IoT, multimedia. Those network slices may vary in security requirement and security policies, while our security orchestrator can adapt itself to provide tenant-specific configurations. In particular, the novel access control paradigm allows different access control models and policies to be specified and enforced according to the particular needs. This use case deserves to further studied.

- **Achieving policy-driven autonomic cyberdefense.** An autonomic cyberdefense system should be able to work seamlessly with the protected assets, achieving self-configuration, self-protection, self-healing, and self-optimiztaion. This is particularly interesting for today's networking infrastructure, which tends to be distributed, large-scale, and intelligent. The design of our security orchestrator demonstrates that a large set of security functions has potential to be implemented and deployed in software-based approaches, significantly improving flexibility and adaptability of security management. We believe this can be treated as an attempt to achieve autonomic cyberdefense, because the key security functions, including monitoring, anomaly detection, and reaction can be systematically integrated together through the specification and enforcement of high-level security policies. However, it is worth pointing out that, the specification, translation, and enforcement of security policies remain as grand challenges in NFV and SDN environments. This challenge holds for both security and networking community, and deserves much more effort from both industry and academia. For example, P4 [24] can be seen as one of the ongoing efforts and promising approaches.

A List of publications

A.1 International journals

1. Montida Pattaranantakul, Ruan He, Qipeng Song, Zonghua Zhang, Ahmed Meddahi, "NFV Security Survey: From Use Case Driven Threat Analysis to State-of-the-Art Countermeasures", *IEEE Communications Surveys and Tutorials*, Vol. 20, No. 4, Jul 2018, pp. 3330–3368.

2. Montida Pattaranantakul, Ruan He, Zonghua Zhang, Ahmed Meddahi, and Ping Wang, "Leveraging Network Functions Virtualization Orchestrators to Achieve Software-Defined Access Control in the Clouds", *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Dec 2018, DOI: 10.1109/TDSC.2018.2889709. (Early access article)

A.2 International conferences and workshops

1. Montida Pattaranantakul, Ruan He, Ahmed Meddahi, and Zonghua Zhang, "SecMANO: Towards Network Functions Virtualization (NFV) based Security MANagement and Orchestration", *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Tianjin, China, Aug 23–26, 2016, pp. 598–605.

2. Montida Pattaranantakul, Yuchia Tseng, Ruan He, Zonghua Zhang, and Ahmed Meddahi, "A First Step Towards Security Extension for NFV Orchestrator", *ACM International Workshop on SDN-NFV Security' 17*, Scottsdale, Arizona, USA, Mar 24, 2017, pp. 25–30. (**Best Paper Award**)

3. Yuchia Tseng, Montida Pattaranantakul, Ruan He, Zonghua Zhang, and Farid Naït-Abdesselam, "Controller DAC: Securing SDN Controller with Dynamic Access Control", *IEEE International Conference on Communications (ICC)*, Paris, France, May 21–25, 2017, pp. 1–6.

4. Ruan He, Montida Pattaranantakul, Zonghua Zhang, and Thomas Duval, "SoDAC: A New Software-Defned Access Control Paradigm for Cloud-based Systems", *The 19th International Conference on Information and Communications Security (ICICS'17)*, Beljing, China, Dec 6-8, 2017, pp. 570–581.

5. Montida Pattaranantakul, Qipeng Song, Yanmei Tian, Zonghua Zhang, Ahmed Meddahi, and Licheng Wang, "Towards Secure and Dependable Service Function Chaining (SFC)", *International Conference on Security and Privacy in Communication Networks (SecureComm' 19)*, Orlando, USA, Oct 23-25, 2019.

A.3 Technical reports

1. Montida Pattaranantakul, Zonghua Zhang, and Ahmed Meddahi, "Network Functions Virtualization (NFV) Security – A Survey", Deliverable for External Research Contract of Orange Labs, delivered on Dec 15, 2015, pp. 1–227.

2. Montida Pattaranantakul, Yuchia Tseng, Zonghua Zhang, and Ahmed Meddahi, "Towards Security Extensions for Cloud Orchestration – A Survey on NFV Management and Orchestration Frameworks", Deliverable for External Research Contract of Orange Labs, delivered on Sep 30, 2016, pp. 1–182.

Bibliography

[1] 3GPP. Telecommunication Management; Study on Management and Orchestration of Network Slicing for Next Generation Network. Technical report, 01 2018.

[2] M. B. A. Bierman and K. Watsen. RESTCONF protocol. https://tools.ietf.org/html/draft-ietf-netconf-restconf-14, Jun 2016.

[3] A. Bierman, and M. Bjorklund, and K. Watsen, and R. Fernando. RESTCONF Protocol. https://www.ietf.org/archive/id/draft-bierman-netconf-restconf-04.txt, Feb 2013.

[4] O. AbdElRahem, A. M. Bahaa-Eldin, and A. Taha. Virtualization security: A survey. In *ICCES*, pages 32–40, Dec 2016.

[5] H. Albaroodi, S. Manickam, and P. Singh. Critical Review of OpenStack Security: Issues and Weaknesses. *Journal of Computer Science*, 10, 2014.

[6] Alcatel Lucent. CloudBand 3.0: The Production Platform for NFV. http://networks.nokia.com/portfolio/solutions/cloudband, Jan 2014. Accessed: 2018-01-11.

[7] ANASTACIA. Advanced Networked Agents for Security and Trust Assessment in CPS/IoT Architectures. http://www.anastacia-h2020.eu/, Jan 2017. Accessed: 2018-01-22.

[8] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati. Supporting Location-based Conditions in Access Control Policies. In *ASIACCS '06*, pages 212–222, 2006.

[9] AT&T Inc. Enhanced Control, Orchestration, Management & Policy Architecture White Paper. https://policyforum.att.com/wp-content/uploads/2017/03/ecomp-architecture-whitepaper-att.pdf, Mar 2016.

[10] A. M. Azab, P. Ning, and X. Zhang. SICE: A Hardware-level Strongly Isolated Computing Environment for x86 Multi-core Platforms. In *CCS '11*, pages 375–388, 2011.

[11] A. Baliga, X. Chen, B. Coskun, G. de los Reyes, S. Lee, S. Mathur, and J. E. Van der Merwe. VPMN: Virtual Private Mobile Network Towards Mobility-as-a-service. In *MCS '11*, pages 7–12, 2011.

[12] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. On Orchestrating Virtual Network Functions. In *CNSM '15*, pages 50–56, 2015.

[13] E. B. Barker, W. C. Barker, W. E. Burr, W. T. Polk, and M. E. Smid. SP 800-57. Recommendation for Key Management, Part 1: General (Revised). Technical report, Gaithersburg, MD, United States, 2007.

[14] M. T. Beck and J. F. Botero. Coordinated Allocation of Service Function Chains. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[15] M. Bellare, C. Namprempre, and G. Neven. Unrestricted Aggregate Signatures. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 411–422, 2007.

[16] S. Berger, R. Caceres, K. Goldman, D. Pendarakis, R. Perez, J. R. Rao, E. Rom, R. Sailer, W. Schildhauer, D. Srinivasan, S. Tal, and E. Valdez. Security for the Cloud Infrastructure: Trusted Virtual Data Center Implementation. *IBM Journal of Research and Development*, 53(4):6:1–6:12, Jul 2009.

[17] S. Berger, R. Cáceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan. TVDc: Managing Security in the Trusted Virtual Datacenter. *SIGOPS Oper. Syst. Rev.*, 42(1):40–47, Jan 2008.

[18] C. Bernardos, A. Rahman, J. Zuniga, L. Contreras, and P. Aranda. Network Virtualization Research Challenges. https://tools.ietf.org/pdf/draft-irtf-nfvrg-gaps-network-virtualization-03.pdf, Oct 2016. Accessed: 2017-10-11.

[19] Big Switch Networks. Project Floodlight. http://www.projectfloodlight.org/, Apr 2016.

[20] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). https://tools.ietf.org/html/rfc6020, Oct 2010.

[21] G. Bloom, E. Leontie, B. Narahari, and R. Simha. *Chapter 12: Hardware and Security - Vulnerabilities and Solutions*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.

[22] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered Multisignatures and Identity-based Sequential Aggregate Signatures, with Applications to Secure Routing. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 276–285, New York, NY, USA, 2007. ACM.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 416–432, 2003.

[24] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[25] J. Brassil. Physical Layer Network Isolation in Multi-tenant Clouds. In *ICDCSW '10*, pages 77–81, June 2010.

[26] Bro. The Bro Network Security Monitor. https://www.bro.org/, Nov 2016. Accessed: 2017-09-23.

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

[28] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry. Practical and Lightweight Domain Isolation on Android. In *SPSM '11*, pages 51–62, 2011.

[29] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow. FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack. In *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pages 109–116, May 2014.

[30] C. Meyer, and J. Schwenk. SoK: Lessons Learned from SSL/TLS Attacks. In *WISA' 13*, pages 189–209, 2014.

[31] G. Carrozzo, R. Szabo, and K. Pentikousis. Network Function Virtualization: Resource Orchestration Challenges. https://tools.ietf.org/pdf/draft-caszpe-nfvrg-orchestration-challenges-00.pdf, Nov 2015. Accessed: 2017-07-16.

[32] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking Control of the Enterprise. In *SIGCOMM '07*, pages 1–12, 2007.

[33] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci. Securing Virtual Network Function Placement with High Availability Guarantees. In *IFIP Networking' 17*, pages 1–9, 2017.

[34] R. Chayapathi, S. F. Hassan, and P. Shah. *Network Functions Virtualization (NFV) with A Touch of SDN*. Pearson Education, 2016.

[35] P.-W. Chi, C.-T. Kuo, J.-W. Guo, and C.-L. Lei. How to detect a compromised SDN switch. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015.

[36] W.-K. Chiang and J.-H. Chen. TW-KEAP: An Efficient Four-party Key Exchange Protocol for End-to-end Communications. In *SIN '11*, pages 167–174, 2011.

[37] T.-S. Chou. Security Threats on Cloud Computing Vulnerabilities. *IJCSIT*, 5, 2013.

[38] C.-Y. Chow and M. F. Mokbel. Privacy in Location-based Services: A System Architecture Perspective. *SIGSPATIAL Special*, 1(2):23–27, July 2009.

[39] A. Cilardo and D. Argenziano. Securing the cloud with reconfigurable computing: An fpga accelerator for homomorphic encryption. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1622–1627, March 2016.

[40] Cisco. Network Virtualization - Path Isolation Design Guide. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Network_Virtualization/PathIsol.pdf, Feb 2009. Accessed: 2017-09-27.

[41] Cisco. Understanding and Configuring VLANs. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/25ew/configuration/guide/conf/vlans.html, Feb 2012. Accessed: 2017-11-18.

[42] Cisco. Virtual Routing and Forwarding. http://www.cisco.com/c/en/us/td/docs/net_mgmt/active_network_abstraction/3-7/reference/guide/ANARefGuide37/vrf.html, Apr 2014. Accessed: 2017-09-27.

[43] Cisco. White Paper: Flexible Workload Mobility and Server Placement with VXLAN Routing on Cisco CSR 1000V and Cisco Nexus 1000V. http://www.cisco.com/c/en/us/products/collateral/routers/cloud-services-router-1000v-series/white-paper-c11-732382.pdf, Aug 2014. Accessed: 2017-11-18.

[44] Cisco. VXLAN Design and Deployment. https://www.cisco.com/c/dam/m/sl_si/events/2016/cisco_dan_inovativnih_resitev/pdf/cisco_day_slovenia_2016_vxlan_marian_klas_final.pdf, Oct 2016. Accessed: 2017-11-18.

[45] Cloud Security Alliance. Cloud Computing Top Threats in 2016. https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf, Feb 2016. Accessed: 2017-07-05.

[46] Cloud Security Alliance. Security Position Paper: Network Function Virtualization. https://downloads.cloudsecurityalliance.org/assets/research/virtualization/Security_Position_Paper-Network_Function_Virtualization.pdf, Feb 2016. Accessed: 2017-08-01.

[47] Cloudify. Cloudify Pure-Play NFV Management and Orchestration. http://cloudify.co/brochures/Cloudify-NFV.pdf, Feb 2015. Accessed: 2018-01-14.

[48] CloudNFV. Taking NFV to the Cloud. http://www.cloudnfv.com/, 2013. Accessed: 2018-01-10.

[49] P. Cox. PaaS Threats In The Cloud. https://systemexperts.com/pdf/SystemExperts-PaaSThreatsInTheCloud.pdf, 2010. Accessed: 2017-07-11.

[50] A. Csoma, B. Sonkoly, L. Csikor, F. Németh, A. Gulyas, W. Tavernier, and S. Sahhaf. ESCAPE: Extensible Service Chain Prototyping Environment Using Mininet, Click, NETCONF and POX. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 125–126, New York, NY, USA, 2014. ACM.

[51] B. Cui and T. Xi. Security Analysis of Openstack Keystone. In *IMIS' 15*, pages 283–288, Jul 2015.

[52] Cybersecurity Working Group. White Paper: Considerations for Securing SDN/NFV. https://transition.fcc.gov/bureaus/oet/tac/tacdocs/reports/2016/Securing%20-SDN-NFV%20-SWG-WP-Final.pdf, Jan 2016. Accessed: 2017-12-23.

[53] W. Dai. Commutative-like Encryption: A New Characterization of ElGamal. *CoRR*, abs/1011.3718, 2010.

[54] G. Davoli, W. Cerroni, C. Contoli, F. Foresta, and F. Callegati. Implementation of service function chaining control plane through OpenFlow. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–4, Nov 2017.

[55] W. Dawoud, I. Takouna, and C. Meinel. Infrastructure as a Service Security: Challenges and Solutions. In *INFOS' 10*, pages 1–8, Mar 2010.

[56] H. Deng, C. Donley, and J. Zemlin. Open-O: Unified NFV/SDN Open Source Orchestrator. https://events.static.linuxfound.org/sites/events/files/slides/ons_plenary_wed_hdeng.pdf, Mar 2016.

[57] J. Deng, H. Hu, H. Li, Z. Pan, K. Wang, G. Ahn, J. Bi, and Y. Park. VNGuard: An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls. In *IEEE NFV-SDN' 15*, pages 107–114, 2015.

[58] J. Deng, H. Li, H. Hu, K. Wang, G. Ahn, Z. Zhao, and W. Han. On the Safety and Efficiency of Virtual Firewall Elasticity Control. In *NDSS' 17*, 2017.

[59] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann. SPHINX: detecting security attacks in software-defined networks. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, pages 1–15, 2015.

[60] A. E. din Mady, R. Trapero, A. Skarmeta, and S. Bianchi. Towards Secure Building Management System based on Internet of Things. In *CENICS' 17*, pages 61–644, Sept 2017.

[61] Docker. Docker Swarm Overview. https://docs.docker.com/swarm/overview/, 2015.

[62] Y. Dodis, W. Luo, S. Xu, and M. Yung. Key-insulated Symmetric Key Cryptography and Mitigating Attacks Against Cryptographic Cloud Software. In *ASIACCS '12*, pages 57–58, 2012.

[63] R. Doriguzzi-Corin, S. Scott-Hayward, D. Siracusa, and E. Salvadori. Application-Centric provisioning of virtual security network functions. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 276–279, Nov 2017.

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

[65] R. A. Eichelberger, T. Ferreto, S. Tandel, and P. A. P. R. Duarte. SFC Path Tracer: A troubleshooting tool for Service Function Chaining. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 568–571, May 2017.

[66] Enisa. Threat Landscape and Good Practice Guide for Software Defined Networks/5G. https://www.enisa.europa.eu/publications/sdn-threat-landscape, Dec 2015. Accessed: 2017-11-03.

[67] ETSI. Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action. http://portal.etsi.org/NFV/NFV_White_Paper.pdf, Oct 2012. Accessed: 2016-06-12.

[68] ETSI. Network Functions Virtualization (NFV); NFV Performance & Portability Best Practices. http://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/001/01.01.02_60/gs_NFV-PER001v010102p.pdf, Dec 2014.

[69] ETSI. Network Functions Virtualization: Ecosystems; Report on SDN Usage in NFV Architectural Framework. https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf, Dec 2015.

[70] ETSI. Network Functions Virtualization (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework. http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf, Dec 2015.

[71] ETSI. Network Functions Virtualization (NFV); Management and Orchestration; Network Service Templates Specification. https://www.etsi.org/deliver/etsi_gs/nfv-ifa/001_099/014/02.01.01_60/gs_nfv-ifa014v020101p.pdf, Oct 2016.

[72] ETSI GR NFV 001. Network Functions Virtualization (NFV); Use Cases. https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20001v1.2.1%20-%20GR%20-%20NFV%20Use%20Cases%20revision.pdf, May 2017. Accessed: 2018-06-03.

[73] ETSI GS NFV 002. Network Functions Virtualization (NFV); Architectural Framework. https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf, Dec 2014. Accessed: 2018-06-05.

[74] ETSI GS NFV-MAN 001. Network Functions Virtualization (NFV); Management and Orchestration. https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-MAN%20001v1.1.1%20-%20GS%20-%20Management%20and%20Orchestration.pdf, Dec 2014. Accessed: 2016-07-10.

[75] ETSI GS NFV-SEC 001. Network Functions Virtualization (NFV); NFV Security; Problem Statement. http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/001/01.01.01_60/gs_NFV-SEC001v010101p.pdf, Oct 2014. Accessed: 2016-07-11.

[76] ETSI GS NFV-SEC 003. Network Functions Virtualization (NFV); NFV Security; Security and Trust Guidance. https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SEC%20003v1.2.1%20-%20GR%20-%20Security%20and%20Trust%20Guidance.pdf, Aug 2016. Accessed: 2018-06-05.

[77] ETSI GS NFV-SEC 006. Network Functions Virtualization (NFV); Security Guide; Report on Security Aspects and Regulatory Concerns. http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/006/01.01.01_60/gs_nfv-sec006v010101p.pdf, Apr 2016. Accessed: 2018-06-01.

[78] ETSI GS NFV-SEC 009. Network Functions Virtualization (NFV); NFV Security; Report on Use cases and Technical Approaches for Multi-layer Host Administration. https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SEC%20009v1.2.1%20-%20GR%20-%20UCs%20for%20multi-layer%20host%20admin.pdf, Jan 2017. Accessed: 2018-06-05.

[79] ETSI GS NFV-SEC 012. Network Functions Virtualization (NFV) Release 3; Security; System Architecture Specification for Execution of Sensitive NFV Components. http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/012/03.01.01_60/gs_NFV-SEC012v030101p.pdf, Jan 2017. Accessed: 2018-06-02.

[80] ETSI GS NFV-SEC 013. Network Functions Virtualization (NFV) Release 3; Security; Security Management and Monitoring Specification. http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/013/03.01.01_60/gs_NFV-SEC013v030101p.pdf, Feb 2017. Accessed: 2018-06-03.

[81] I. Farris, J. B. Bernabe, N. Toumi, D. Garcia-Carrillo, T. Taleb, A. Skarmeta, and B. Sahlin. Towards Provisioning of SDN/NFV-based Security Enablers for Integrated Protection of IoT Systems. In *CSCN' 17*, pages 169–174, Sept 2017.

[82] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey. Bohatei: Flexible and Elastic DDoS Defense. In *SEC' 15*, pages 817–832, 2015.

[83] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-based Access Control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, Aug 2001.

[84] M. D. Firoozjaei, J. Jeong, H. Ko, and H. Kim. Security Challenges with Network Functions Virtualization. *Future Generation Comp. Syst.*, Feb 2017.

[85] M. D. Firoozjaei, J. P. Jeong, H. Ko, and H. Kim. Security Challenges with Network Functions Virtualization. *Future Generation Computer Systems*, 67:315–324, 2017.

[86] M. Flittner, J. M. Scheuermann, and R. Bauer. ChainGuard: Controller-independent verification of service function chaining in cloud computing. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7, Nov 2017.

[87] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proc. Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[88] A. Gember, R. Grandl, J. Khalid, and A. Akella. Design and Implementation of a Framework for Software-defined Middlebox Networking. *SIGCOMM Comput. Commun. Rev.*, 43(4):467–468, Aug. 2013.

[89] S. Gharout, A. Bouabdallah, M. Kellil, and Y. Challal. Key Management with Host Mobility in Dynamic Groups. In *SIN '10*, pages 186–194, 2010.

[90] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba. Service Function Chaining Simplified. *CoRR*, abs/1601.00751, 2016.

[91] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba. Distributed Service Function Chaining. *IEEE Journal on Selected Areas in Communications*, 35(11):2479–2489, Nov 2017.

[92] K. Giotis, Y. Kryftis, and V. Maglaris. Policy-based Orchestration of NFV Services in Software-Defined Networks. In *NetSoft' 15*, pages 1–5, Apr 2015.

[93] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard. Dependability of the NFV Orchestrator: State of the Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 20(4):3307–3329, Fourthquarter 2018.

[94] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an Operating System for Networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110, Jul 2008.

[95] W. Haeffner, J. Napper, M. Stiemerling, and D. Lopez. Service Function Chaining Use Cases in Mobile Networks. https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-09, Jan 2019.

[96] J. Halpern and C. Pignataro. Service Function Chainning (SFC) Architecture. https://tools.ietf.org/html/rfc7665, Oct 2015.

[97] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3):134–141, Mar 2006.

[98] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network Function Virtualization: Challenges and Opportunities for Innovations. *IEEE Communications Magazine*, 53(2):90–97, Feb 2015.

[99] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2005.

[100] D. Hardt. The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749, Oct 2012. Accessed: 2017-02-16.

[101] M. Hashimoto and J. Bender. Vagrant. https://www.vagrantup.com/, Mar 2010.

[102] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernández. An Analysis of Security Issues for Cloud Computing. *J. Internet Services and Applications*, 4(1):5:1–5:13, 2013.

[103] R. He, M. Pattaranantakul, Z. Zhang, and T. Duval. SDAC: A New Software-Defined Access Control Paradigm for Cloud-Based Systems. In *ICICS' 17*, pages 570–581, 2017.

[104] Heat. OpenStack Orchestration Service. https://wiki.openstack.org/wiki/Heat, May 2014.

[105] S. Hong, L. Xu, H. Wang, and G. Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, pages 1–15, 2015.

[106] HostAP. Hostapd: IEEE 802.11 AP, IEEE 802.1X/ WPA/ WPA2/ EAP/ RADIUS Authenticator. http://w1.fi/hostapd/, Jan 2012. Accessed: 2017-02-15.

[107] HP. Business White Paper: The Reality of Cost Reduction (Rev. 2). https://www.hpe.com/h20195/v2/getpdf.aspx/4AA5-2160ENW.pdf, Apr 2017. Accessed: 2017-11-06.

[108] H. Huang, S. Guo, J. Wu, and J. Li. Service Chaining for Hybrid Network Function. *IEEE Transactions on Cloud Computing*, pages 1–1, 2017.

[109] Y. L. Huang, B. Chen, M. W. Shih, and C. Y. Lai. Security Impacts of Virtualization on a Network Testbed. In *SERE' 12*, pages 71–77, Jun 2012.

[110] Huawei. White Paper - Observation to NFV. http://www.huawei.com/ilink/en/download/HW_399662, 2014. Accessed: 2017-06-10.

[111] J. Y. Hwang, D. H. Lee, and M. Yung. Universal Forgery of the Identity-based Sequential Aggregate Signature Scheme. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 157–160, New York, NY, USA, 2009. ACM.

[112] A. S. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy. CloudSec: A Security Monitoring Appliance for Virtual Machines in the IaaS Cloud Model. In *NSS '11*, pages 113–120, Sep 2011.

[113] IHS Markit. Network Functions Virtualization Market Worth Over 15B Dollars By 2020. http://news.ihsmarkit.com/press-release/technology/network-functions-virtualization-market-worth-over-15-billion-2020-says-ihs, Jul 2016. Accessed: 2017-09-07.

[114] Infonetics Research. The Evolution of SDN and NFV Orchestration. https://www.juniper.net/assets/cn/zh/local/pdf/analyst-reports/2000604-en.pdf, Feb 2015. Accessed: 2017-05-24.

[115] E. Jacob, J. Matias, A. Mendiola, V. Fuentes, J. Garay, and C. Pinedo. Deploying a Virtual Network Function over a Software Defined Network Infrastructure: Experiences Deploying an Access Control VNF in the University of Basque Country's OpenFlow Enabled Eacility. In *TNC '14*, May 2014.

[116] B. Jaeger. Security Orchestrator: Introducing a Security Orchestrator in the Context of the ETSI NFV Reference Architecture. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1255–1260, Aug 2015.

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

[118] S. Jain, F. Shafique, V. Djeric, and A. Goel. Application-level Isolation and Recovery with Solitude. *SIGOPS Oper. Syst. Rev.*, 42(4):95–107, Apr 2008.

[119] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park. Kargus: A Highly-scalable Software-based Intrusion Detection System. In *CCS '12*, pages 317–328, 2012.

[120] X. Jin, R. Krishnan, and R. Sandhu. A Unified Attribute-based Access Control Model Covering DAC, MAC and RBAC. In *DBSec' 12*, pages 41–55, Berlin, Heidelberg, 2012. Springer-Verlag.

[121] Y. Juba, H.-H. Huang, and K. Kawagoe. POSTER: Security Control System Enabling to Keep an Intra-LAN in a Secure State Using Security-and-Performance Ratio Control Policies. In *CCS '14*, pages 1442–1444, 2014.

[122] A. A. E. Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization Based Access Control. In *POLICY' 03*, pages 120–131, Washington, DC, USA, 2003. IEEE Computer Society.

[123] J. Keeney, S. v. d. Meer, and L. Fallon. Towards Real-time Management of Virtualized Telecommunication Networks. In *CNSM' 14*, pages 388–393, Nov 2014.

[124] A. Kern and C. Walhorn. Rule Support for Role-based Access Control. In *SACMAT '05*, pages 130–138, 2005.

[125] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig. Lightweight Source Authentication and Path Validation. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 271–282, New York, NY, USA, 2014. ACM.

[126] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[127] S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma. Service Function Chaining Use Cases in Data Centers. https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06, Feb 2017.

[128] S. Lal, T. Taleb, and A. Dutta. NFV: Security Threats and Best Practices. *IEEE Communications Magazine*, PP(99):2–8, May 2017.

[129] K. E. Lauter. Practical Applications of Homomorphic Encryption. In *CCSW' 12*, pages 57–58, 2012.

[130] A. Lemke. How to Manage Security in NFV Environment. https://insight.nokia.com/how-manage-security-nfv-environments, Dec 2014. Accessed: 2017-12-22.

[131] Q. Li, X. Zhang, M. Xu, and J. Wu. Towards secure dynamic collaborations with group-based RBAC model. *Computers & Security*, 28(5):260–275, Jul 2009.

[132] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee. Dynamic Packet Forwarding Verification in SDN. *IEEE Transactions on Dependable and Secure Computing*, pages 1–16, 2018.

[133] Y. Li and M. Chen. Software-Defined Network Function Virtualization: A Survey. *IEEE Access*, 3:2542–2553, 2015.

[134] Linux Foundation. OVS: Open vSwitch. https://www.openvswitch.org/, Feb 2019.

[135] Linux Foundation Projects. Open Security Controller. https://www.opensecuritycontroller.org/, Feb 2017.

[136] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Z. Cao, J. Hu, and C. Pham. Service Function Chaining (SFC) General Use Cases. https://tools.ietf.org/html/draft-liu-sfc-use-cases-08, Sep 2014.

[137] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106, May 2015.

[138] B. Lynn. PBC (Pairing-Based Cryptography) Library. https://crypto.stanford.edu/pbc/, 2006.

[139] T. Macaulay. The 7 Deadly Threats to 4G: 4G LTE Security Roadmap and Reference Design. http://www.webtorials.com/main/resource/papers/McAfee/paper20/7-deadly-threats-4g.pdf, 2013. Accessed: 2017-07-25.

[140] C. Makaya, D. Freimuth, D. Wood, and S. Calo. Policy-based NFV Management and Orchestration. In *NFV-SDN' 15*, pages 128–134, Nov 2015.

[141] E. Markatos. Large Scale Attacks on the Internet Lessons learned from the LOBSTER Project. https://www.ist-lobster.org/publications/presentations/markatos-attacks.pdf, Jun 2008. Accessed: 2017-07-16.

[142] B. Martini and F. Paganelli. A Service-Oriented Approach for Dynamic Chaining of Virtual Network Functions over Multi-Provider Software-Defined Networks. *Future Internet*, 8(2), 2016.

[143] S. Mavoungou, G. Kaddoum, M. Taha, and G. Matar. Survey on Threats and Attacks on Mobile Networks. *IEEE Access*, 4:4543–4572, Sep 2016.

[144] J. McDowall. Inherent Security Design Patterns for SDN/NFV Deployments. http://events.linuxfoundation.org/sites/events/files/slides/Design_Patterns_submitted.pdf, 2016. Accessed: 2017-10-11.

[145] D. McKay. A Deep Dive Into Hyperjacking. http://www.securityweek.com/deep-dive-hyperjacking, Mar 2011. Accessed: 2017-02-25.

[146] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache. NFV Orchestration Framework Addressing SFC Challenges. *IEEE Communications Magazine*, 55(6):16–23, Jun 2017.

[147] M. Mechtri, C. Ghribi, and D. Zeghlache. A Scalable Algorithm for the Placement of Service Function Chains. *IEEE Trans. on Netw. and Serv. Manag.*, 13(3):533–546, Sep 2016.

[148] A. M. Medhat, G. A. Carella, M. Pauls, and T. Magedanz. Extensible framework for elastic orchestration of service function chains in 5G networks. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 327–333, Nov 2017.

[149] A. M. Medhat, G. A. Carella, M. Pauls, M. Monachesi, M. Corici, and T. Magedanz. Resilient Orchestration of Service Functions Chains in a NFV environment. In *NFV-SDN' 16*, pages 7–12, Nov 2016.

[150] S. Meng and L. Liu. Monitoring-as-a-service in the Cloud: Spec Phd Award (Invited Abstract). In *ICPE '13*, pages 373–374, 2013.

[151] C. Meyer and J. Schwenk. Lessons Learned From Previous SSL/TLS Attacks - A Brief Chronology Of Attacks And Weaknesses. Cryptology ePrint Archive, Report 2013/049, 2013.

[152] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, and D. Lopez. Management and Orchestration Challenges in Network Functions Virtualization. *IEEE Communications Magazine*, 54(1):98–105, Jan 2016.

[153] C. Modi, D. Patel, B. Borisanya, A. Patel, and M. Rajarajan. A Novel Framework for Intrusion Detection in Cloud. In *SIN '12*, pages 67–74, 2012.

[154] A. Morais and A. Cavalli. A Distributed Intrusion Detection Scheme for Wireless Ad Hoc Networks. In *SAC '12*, pages 556–562, 2012.

[155] M. A. Morsy, J. Grundy, and I. Müller. An Analysis of The Cloud Computing Security Problem. In *APSEC '10*, pages 1–6, Nov 2010.

[156] J. Myerson. Protect a PaaS from Hackers with Four Phases of Defense. http://www.techrepublic.com/article/protect-a-paas-from-hackers-with-four-phases-of-defense/, Nov 2014. Accessed: 2017-10-23.

[157] Neutron. OpenStack Networking Service. https://wiki.openstack.org/wiki/Neutron, Aug 2017.

[158] V. Nguyen, A. Brunström, K. Grinnemo, and J. Taheri. SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey. *IEEE Communications Surveys and Tutorials*, 19(3):1567–1602, 2017.

[159] NTT Labs. RYU SDN Framework. https://osrg.github.io/ryu/, 2016.

[160] R. V. Nunes, R. L. Pontes, and D. Guedes. Virtualized Network Isolation using Software Defined Networks. In *LCN '13*, pages 683–686, Oct 2013.

[161] OASIS. Security Assertion Markup Language (SAML) V2.0 Technical Overview. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf, Mar 2014. Accessed: 2017-10-21.

[162] J. Oberheide, E. Cooke, and F. Jahanian. CloudAV: N-version Antivirus in the Network Cloud. In *SS' 08*, pages 91–106, 2008.

[163] ONAP. Open Network Automation Platform. https://www.onap.org/, Apr 2017. Accessed: 2018-01-17.

[164] Open Networking Foundation. OpenFlow Switch Specification. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf, Jun 2012.

[165] Open Networking Foundation. OpenFlow-enabled SDN and Network Functions Virtualization. https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf, Feb 2014.

[166] Open Networking Foundation. SDN Architecture Overview. https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf, Jun 2014.

[167] Open Networking Foundation. Relationship of SDN and NFV. https://www.opennetworking.org/wp-content/uploads/2014/10/onf2015.310_Architectural_comparison.08-2.pdf, Oct 2015.

[168] OpenBaton. OpenBaton is a ETSI NFV Compliant Management and Orchestration (MANO) Framework. https://openbaton.github.io/, Jan 2016. Accessed: 2018-01-15.

[169] OpenID. Welcome to OpenID Connect. http://openid.net/connect/, Feb 2014. Accessed: 2017-02-20.

[170] OpenStack. Open Source Software for Creating Private and Public Clouds. https://www.openstack.org/, Oct 2016. Accessed: 2018-02-12.

[171] OpenStack. Keystone - OpenStack Identity Service. https://docs.openstack.org/keystone/pike/, Oct 2017. Accessed: 2018-01-18.

[172] OpenStack. OpenStack Compute (Nova). https://docs.openstack.org/nova/latest/, Mar 2019. Accessed: 2019-04-02.

[173] OPNFV. Moon - Security Management Module. https://wiki.opnfv.org/display/moon/Moon, Apr 2016. Accessed: 2018-01-17.

[174] OPNFV. Open Plaform for NFV Project. https://wiki.opnfv.org, Jan 2017. Accessed: 2018-02-12.

[175] Oracle Coorporation. VirtualBox. https://www.virtualbox.org/, Jan 2007.

[176] P. Paganini. Hardware Attacks, Backdoors, and Electronic Component Qualification. http://resources.infosecinstitute.com/hardware-attacks-backdoors-and-electronic-component-qualification/, Oct 2013. Accessed: 2017-10-28.

[177] D. H. Parekh and R. Sridaran. An Analysis of Security Challenges in Cloud Computing. *Journal of Advanced Computer Science and Applications*, 14, 2013.

[178] M. Pattaranantakul, R. He, A. Meddahi, and Z. Zhang. SecMANO: Towards Network Functions Virtualization (NFV) Based Security MANagement and Orchestration. In *IEEE Trustcom/BigDataSE/ISPA' 16*, pages 598–605, 2016.

[179] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi. NFV Security Survey: From Use Case Driven Threat Analysis to State-of-the-Art Countermeasures. *IEEE Communications Surveys and Tutorials*, 20(4):3330–3368, 2018.

[180] M. Pattaranantakul, Y. Tseng, R. He, Z. Zhang, and A. Meddahi. A First Step Towards Security Extension for NFV Orchestrator. In *ACM International Workshop on SDN-NFVSec@CODASPY' 17*, pages 25–30, 2017.

[181] G. Pék, L. Buttyán, and B. Bencsáth. A Survey of Security Issues in Hardware Virtualization. *ACM Comput. Surv.*, 45(3):40:1–40:34, Jul 2013.

[182] Pica 8, Inc. OpenFlow-enabled Ethernet Switches. https://docs.pica8.com/, Aug 2009.

[183] POX. POX Wiki. https://openflow.stanford.edu/display/ONL/POX+Wiki, Mar 2015. Accessed: 2017-10-03.

[184] C. Priebe, D. Muthukumaran, D. O' Keeffe, D. Eyers, B. Shand, R. Kapitza, and P. Pietzuch. CloudSafetyNet: Detecting Data Leakage Between Cloud Tenants. In *CCSW '14*, pages 117–128, 2014.

[185] PyPBC. Python Binding for PBC. https://github.com/debatem1/pypbc, Nov 2017.

[186] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. *SIGCOMM Comput. Commun. Rev.*, 43(4):27–38, Aug 2013.

[187] Qihoo. Virtual Machine Escape Fetches 105,000 dollars at Pwn20wn Hacking Contest. https://arstechnica.com/information-technology/2017/03/hack-that-escapes-vm-by-exploiting-edge-browser-fetches-105000-at-pwn2own/, Mar 2017. Accessed: 2017-12-12.

[188] P. Quinn, U. Elzur, and C. Pignataro. Network Service Hearder (NSH). https://www.rfc-editor.org/rfc/pdfrfc/rfc8300.txt.pdf, Jan 2018.

[189] P. Quinn and T. Nadeau. Problem Statement for Service Function Chaining. https://tools.ietf.org/html/rfc7498#page-6, Apr 2015.

[190] J. S. R. Enns, M. Bjorklund and A. Bierman. Network Configuration Protocol (NETCONF. https://tools.ietf.org/html/rfc6241, Jun 2011.

[191] R. Mijumbi and J. Serrat and J. L. Gorricho and N. Bouten and F. De Turck and R. Boutaba. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, Feb 2016.

[192] RabbitMQ. RabbitMQ: Open Source Message Broker Software. http://www.rabbitmq.com/, March 2016.

[193] M. O. Rabin. How To Exchange Secrets with Oblivious Transfer. http://eprint.iacr.org/2005/187, 2005. Accessed: 2017-12-03.

[194] Radiant Logic. Toward a Federated Identity Service Based on Virutalization. http://www.radiantlogic.com/learning-center/product-downloads/, 2014. Accessed: 2017-11-02.

[195] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. https://tools.ietf.org/html/rfc3168, Sep 2001.

[196] G. Ranga and S. Flowerday. Identity and Access Management for the Distribution of Social Grants in South Africa. In *SAICSIT '07*, pages 125–131, 2007.

[197] P. Ranjith, C. Priya, and K. Shalini. On Covert Channels Between Virtual Machines. *Journal in Computer Virology*, 8, 2012.

[198] S. Ravidas, S. Lal, I. Oliver, and L. Hippelainen. Incorporating Trust in NFV: Addressing the Challenges. In *ICIN' 17*, pages 87–91, March 2017.

[199] F. Reynaud, F. X. Aguessy, O. Bettan, M. Bouet, and V. Conan. Attacks against Network Functions Virtualization and Software-Defined Networking: State-of-the-art. In *NetSoft' 16*, pages 471–476, June 2016.

[200] M. Roesch. Snort. https://www.snort.org/, Dec 2014. Accessed: 2017-09-24.

[201] S. A. Rouiller. Virtual LAN Security: Weaknesses and countermeasures. https://www.sans.org/reading-room/whitepapers/networkdevs/virtual-lan-security-weaknesses-countermeasures-1090, 2012. Accessed: 2017-09-08.

[202] C. E. Rubio-Medrano, Z. Zhao, A. Doupe, and G.-J. Ahn. Federated Access Management for Collaborative Network Environments: Framework and Case Study. In *SACMAT '15*, pages 125–134, 2015.

[203] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. V. Doorn, J. L. Griffin, and S. Berger. sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. In *IBM Research Report RC23511*, 2005.

[204] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based Access Control Models. *IEEE Computer*, 29(2):38–47, Feb 1996.

[205] T. Sasaki, C. Pappas, T. Lee, T. Hoefler, and A. Perrig. SDNsec: Forwarding Accountability for the SDN Data Plane. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10, Aug 2016.

[206] S. Scott-Hayward, S. Natarajan, and S. Sezer. A survey of security in software defined networks. *IEEE Communications Surveys Tutorials*, 18(1):623–654, Firstquarter 2016.

[207] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and Implementation of a Consolidated Middlebox Architecture. In *NSDI'12*, pages 24–24, 2012.

[208] A. S. Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet. Efficient Provisioning of Security Service Function Chaining Using Network Security Defense Patterns. *IEEE Transactions on Services Computing*, pages 1–1, 2017.

[209] P. K. Shanmugam, N. D. Subramanyam, J. Breen, C. Roach, and J. Van der Merwe. DEIDtect: Towards Distributed Elastic Intrusion Detection. In *DCC '14*, pages 17–24, 2014.

[210] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making Middleboxes Someone else's Problem: Network Processing As a Cloud Service. In *SIGCOMM '12*, pages 13–24, 2012.

[211] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. BlindBox: Deep Packet Inspection over Encrypted Traffic. *SIGCOMM Comput. Commun. Rev.*, 45(4):213–226, Aug 2015.

[212] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson. FRESCO: Modular Composable Security Services for Software-Defined Networks. In *NDSS' 13*, Feb 2013.

[213] S. Shin, H. Wang, and G. Gu. A First Step Toward Network Security Virtualization: From Concept To Prototype. *IEEE Transactions on Information Forensics and Security*, 10(10):2236–2249, Oct 2015.

[214] Y. Sim and H. Y. Lee. Poster: Denial-of-service attack using host location hijacking in software-defined network. In *1st IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 1–2, Mar 2016.

[215] SNS Telecom. The SDN, NFV, Network Virtualization Ecosystem: 2016 - 2030 - Opportunities, Challenges, Strategies, Forcasts. http://www.snstelecom.com/sdn-nfv, Jan 2018. Accessed: 2018-02-22.

[216] Y. Song, H. Kim, and A. Mohaisen. A Private Walk in the Clouds: Using End-to-End Encryption between Cloud Applications in a Personal Domain. In *Lecture Notes in Computer Science*, volume 8647, pages 72–82, 2014.

[217] I. Studnia, E. Alata, Y. Deswarte, M. Kaaniche, and V. Nicomette. Survey of Security Problems in Cloud Computing Virtual Machines. In *C&ESAR' 12*, pages 61–74, Nov 2012.

[218] S. Suzuki and S. Kondo. Dynamic Network Separation for IPv6 Network Security Enhancement. In *SAINT Workshops' 05*, pages 22–25, Jan 2005.

[219] Tacker. Tacker - OpenStack NFV Orchestration. https://wiki.openstack.org/wiki/Tacker, 2013. Accessed: 2018-01-20.

[220] H. Takabi. Privacy Aware Access Control for Data Sharing in Cloud Computing Environments. In *SCC '14*, pages 27–34, 2014.

[221] H. Takabi, J. B. D. Joshi, and G.-J. Ahn. Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security and Privacy*, 8(6):24–31, Nov 2010.

[222] S. Teerakanok, C. Vorakulpipat, and S. Kamolphiwong. Anonymity Preserving Framework for Location-based Information Services. In *MEDES '10*, pages 107–113, 2010.

[223] Telefonica. OpenMANO. http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab/openmano, Apr 2015. Accessed: 2018-01-16.

[224] The Linux Foundation. ONOS (Open Network Operating System). https://wiki.onosproject.org/display/ONOS/Wiki+Home, Dec 2014.

[225] The Linux Foundation Projects. OpenDayLight Project. https://www.opendaylight.org/, Apr 2013.

[226] The Linux Foundation Projects. OpenDaylight Flurine Release. https://www.opendaylight.org/what-we-do/current-release/fluorine, Aug 2018.

[227] TOSCA. Simple Profile for Network Functions Virtualization (NFV version 1.0). http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.pdf, Mar 2016. Accessed: 2018-01-20.

[228] B. Tschaen, Y. Zhang, T. Benson, S. Banerjee, J. Lee, and J. Kang. SFC-Checker: Checking the correct forwarding behavior of Service Function chaining. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 134–140, Nov 2016.

[229] V. Varadharajan and U. Tupakula. Securing Services in Networked Cloud Infrastructures. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2016.

[230] M. Veeraraghavan, T. Sato, M. Buchanan, R. Rahimi, S. Okamoto, and N. Yamanaka. Network Function Virtualization: A Survey. *IEICE Transactions on Communications*, 100(11):1978–1991, 2017.

[231] VMware. vShield Installation and Upgrade Guide. http://www.vmware.com/pdf/vshield_55_install.pdf, 2015. Accessed: 2017-10-01.

[232] A. Vu and Y. Kim. An implementation of hierarchical service function chaining using OpenDaylight platform. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 411–416, Jun 2016.

[233] E. Wang, K. Leung, J. Felix, J. Lyer, and P. Patel. Service Function Chaining Use Cases for Network Security. https://tools.ietf.org/html/draft-wang-sfc-ns-use-cases-03, Jun 2017.

[234] WPA. Linux WPA/WPA2/IEEE 802.1X Supplicant. http://w1.fi/wpa_supplicant/, Jan 2012. Accessed: 2017-10-15.

[235] C. Wraight. White Paper: Content-Aware Identity & Access Management in a Virtual Environment. http://marketing.computerworld.com/content_aware_iam_june2010.pdf, Jun 2010. Accessed: 2017-10-27.

[236] C. Wueest, M. B. Barcena, and L. O'Brien. Mistakes in the IaaS Cloud Could Put Your Data At Risk. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/mistakes-in-the-iaas-cloud-could-put-your-data-at-risk.pdf, May 2015. Accessed: 2017-07-21.

[237] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51, Firstquarter 2015.

[238] G. Xilouris. Overall System Architecture and Interfaces: Version 1.0. http://www.t-nova.eu/wp-content/uploads/2016/03/TNOVA_D2.22_Overall_System_Architecture_and_Interfaces_v1.0.pdf, Sep 2015. Accessed: 2018-01-15.

[239] J. Xiong, Z. Yao, J. Ma, X. Liu, Q. Li, and T. Zhang. PRAM: Privacy Preserving Access Management Scheme in Cloud Services. In *Cloud Computing '13*, pages 41–46, 2013.

[240] K. Yang, X. Jia, and K. Ren. Attribute-based Fine-grained Access Control with Efficient Revocation in Cloud Storage Systems. In *ASIA CCS '13*, pages 523–528, 2013.

[241] W. Yang and C. Fung. A Survey on Security in Network Functions Virtualization. In *NetSoft' 16*, pages 15–19, Jun 2016.

[242] Y. Yang, X. Chen, G. Wang, and L. Cao. An Identity and Access Management Architecture in Cloud. In *ISCID '14*, pages 200–203, Dec 2014.

[243] A. C.-C. Yao. How to Generate and Exchange Secrets. In *SFCS '86*, pages 162–167, 1986.

[244] F. Yao, R. Sprabery, and R. H. Campbell. CryptVMI: A Flexible and Encrypted Virtual Machine Introspection System in the Cloud. In *SCC '14*, pages 11–18, 2014.

[245] S. S. Yau and H. G. An. Protection of Users' Data Confidentiality in Cloud Computing. In *Internetware '10*, pages 11:1–11:6, 2010.

[246] N. Zhang, H. Li, H. Hu, and Y. Park. Towards Effective Virtualization of Intrusion Detection Systems. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, SDN-NFVSec '17, pages 47–50, New York, NY, USA, 2017. ACM.

[247] P. Zhang. Towards Rule Enforcement Verification for Software Defined Networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[248] T. Zhang and R. B. Lee. CloudMonatt: An Architecture for Security Health Monitoring and Attestation of Virtual Machines in Cloud Computing. In *ISCA' 15*, pages 362–374, Jun 2015.

[249] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula. StEERING: A software-defined networking for inline service chaining. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–10, Oct 2013.

[250] Y. Zhang, W. Lee, and Y.-A. Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *Wirel. Netw.*, 9(5):545–556, Sep 2003.

[251] Y. Zhang, W. Wu, S. Banerjee, J. Kang, and M. A. Sanchez. SLA-verifier: Stateful and quantitative verification for service chaining. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[252] G. Zhong and U. Hengartner. Toward a Distributed K-anonymity Protocol for Location Privacy. In *WPES' 08*, pages 33–38, 2008.

[253] D. Zou, W. Zhang, W. Qiang, G. Xiang, L. T. Yang, H. Jin, and K. Hu. Design and Implementation of a Trusted Monitoring Framework for Cloud Platforms. *Future Gener. Comput. Syst.*, 29(8):2092–2102, Oct 2013.

[254] L. Zuccaro, F. Cimorelli, F. D. Priscoli, C. G. Giorgi, S. Monaco, and V. Suraci. Distributed Control in Virtualized Networks. *Procedia Computer Science*, 56:276–283, 2015.