Software-defined security for network function virtualization

Manel Smine

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÈRIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Manel SMINE

Software-defined Security for Network Function Virtualization

Thèse présentée et soutenue à Cesson Sévigné à IMT Atlantique, le 05/12/2022
Unité de recherche : LabSTICC
Thèse N° : 2022IMTA0323

Rapporteurs avant soutenance :

Joaquin Garcia-Alfaro	Professeur Télécom SudParis
Rémi Badonnel	Professeur des Universités Université de Lorraine

Composition du Jury :

Président :	Abdelmalek Benzekri	Professeur des universités, Université Paul Sabatier Toulouse III
Examinateurs :	Joaquin Garcia-Alfaro	Professeur Télécom SudParis
	Rémi Badonnel	Professeur des Universités Université de Lorraine
	Françoise Sailhan	Professeure, IMT-Atlantique
Dir. de thèse :	David Espes	Professeur, Université de Bretagne Occidentale
Encadrant de thèse :	Marc-Oliver Pahl	Directeur de recherche, IMT Atlantique

Dedicated to my parents, my husband, my two brothers, my sister and my two children for 1001 reasons.

ABSTRACT

Network Function Virtualization (NFV) technology is a concept that aims to virtualize basic network functions that are traditionally performed by physical elements, such as routing. This concept has been proposed to improve the flexibility of deployment and cost effectiveness of network services by eliminating the problems imposed by manual processes. Also it allows better sharing of information between their components. In particular, NFV enables a cost-effective transformation from hardware-based to software-defined services. In addition, it allows the deployment of new services on demand. Despite the above-mentioned advantages, existing NFV infrastructures suffer from several security issues such as security breaches, data loss and information leakage that can lead to serious security threats such as user privacy violation and/or exposure of confidential data. To mitigate these threats, many security mechanisms have been proposed in the literature such as intrusion detection and prevention, access and flow control and data encryption, etc. Particularly, once well-specified and correctly-enforced, access control policies can help reduce many security threats such as, DoS attack, data leakage, breach of isolation, cross VM Attack, malicious insiders, etc.

The main objective of this thesis is to improve the security of NFV services through access control enforcement. We start by studying the existing and potential security problems and threats in NFV infrastructures. Then, we propose a taxonomy for classifying identified threats according to the components that are affected by these threats. Afterwards, we review existing solutions for enforcing access control policies in NFV infrastructures and provide a comprehensive comparative overview of existing solutions with respect to several properties, including the considered adversary model, effectiveness, correctness, optimality, etc. The conducted comparative overview allows us to identify several open challenges that were addressed through the following contributions.

In a first step, we define an expressive specification model to be able to express high-level access control requirements to be enforced over network services. Then, we show that our specification model can correctly express access control requirements

specified using well known access control models such as RBAC, ABAC, and ORBAC. Second, we propose a provably correct method for refining high-level access control requirements towards a Domain Type Enforcement (DTE) concrete specification. Then, we define an ETSI-NFV compliant, efficient, and scalable enforcement method of access control policies in NFV services.

In a second step, we extend the proposed model to enable a clean and efficient deployment of complex access control policies containing exceptions and/or conflicting rules on NFV services. Experimental results show that, compared to similar existing solutions, our model significantly reduces (1) the complexity of the DTE specification to be deployed and (2) the network performance impact of the NFV service in which the policies are deployed.

In a third step, we consider a strong adversary model in which an insider adversary is controlling one or more unknown nodes (Virtual Network Functions) that compose the NFV service. We propose a correctly provable access control policy enforcement model that computes optimally the right set of rules that should be deployed in each network link connecting two VNFs . This model makes violating the enforced access control requirement impossible for the insider adversary.

In a fourth step, we aim to find the best trade-off between the resources required for the access control policy deployment and the impact on the quality of NFV service due to the enforcement of access control requirements. We propose a model to quantify the impact in terms of latency of the deployment of the access control model on the target NFV service. Then, we define a minimization problem that aims to minimize both the impact in terms of latency in an NFV service and the computation and storage resources that are required by the access control policy enforcement points. Afterwards, we show that the proposed optimization problem is nonlinear and non-convex and we use an improvement of the Non-dominated Sorting Genetic Algorithm NSGA II for solving it. Finally, we build a simulation framework to evaluate the effectiveness of the proposed optimization model. The conducted simulations show that the optimization model we propose allows security administrators to have enough information to make decisions on the right trade-off between the quality of service degradation and the quantity of resources that should be dedicated to the access control policy enforcement points.

RÉSUMÉ

La nouvelle technologie NFV (Network Function Virtualization) est un concept qui vise à virtualiser les fonctions de base du réseau qui sont traditionnellement réalisées par des éléments physiques, comme le routage. Ce concept a été proposé pour améliorer la flexibilité du déploiement et la rentabilité des services de réseau en éliminant les problèmes imposés par les processus manuels. Aussi, elle permet un meilleur partage des informations entre leurs composants. En particulier, NFV permet une transformation rentable des services basés sur le matériel vers des services définis par logiciel. En outre, elle permet le déploiement de nouveaux services à la demande. Malgré les avantages susmentionnés, les infrastructures NFV existantes souffrent de plusieurs problèmes de sécurité tels que les failles de sécurité, les pertes de données et les fuites d'informations qui peuvent entraîner de graves menaces pour la sécurité, comme la violation de la vie privée des utilisateurs et/ou l'exposition de données confidentielles. Pour atténuer ces menaces, de nombreux mécanismes de sécurité ont été proposés dans la littérature, tels que la détection et la prévention des intrusions, le contrôle d'accès et de flux, le cryptage des données, etc.

L'objectif de cette thèse est d'améliorer la sécurité des services NFVs. Pour atteindre cet objectif, nous avons proposé un certain nombre de contributions. Tout d'abord, nous avons étudié les problèmes et les menaces de sécurité existants et potentiels dans les infrastructures NFVs. Ensuite, nous les avons classés en fonction des composants qui sont affectés par ces menaces. Puis, nous avons étudié les différents mécanismes de sécurité qui peuvent être utilisés pour réduire ces risques. Cette étude nous a permis de réaliser que le déploiement des politiques de contrôle d'accès au niveau des services NFV permet d'atténuer plusieurs problèmes de sécurité tels que, les vulnérabilités des VMs invitées, les utilisateurs malveillants, les fuites d'informations, etc. Ce constat nous a conduit à examiner les solutions existantes pour le déploiement des politiques de contrôle d'accès dans les infrastructures NFV. Troisièmement, nous fournissons un aperçu comparatif complet des solutions existantes en se basant sur plusieurs propriétés, y compris le modèle d'adversaire considéré, l'efficacité, la véracité, l'optimalité, etc. L'aperçu comparatif réalisé nous a permis

d'identifier plusieurs défis ouverts qui ont été abordés aux travers des contributions suivantes.

Dans un premier temps, nous définissons un modèle de spécification expressif permettant d'exprimer les exigences de contrôle d'accès de haut niveau à déployer aux services de réseau. Nous montrons ensuite que notre modèle de spécification peut exprimer correctement les exigences de contrôle d'accès spécifiées à l'aide de plusieurs modèles de contrôle d'accès bien connus tels que RBAC, ABAC et ORBAC. Ensuite, nous proposons une méthode prouvée correcte pour raffiner les exigences de contrôle d'accès de haut niveau vers une spécification concrète de type et de domaine (DTE). Enfin, nous définissons une méthode de déploiement des politiques de contrôle d'accès dans les services NFV qui soit conforme à la norme ETSI-NFV, efficace et évolutive.

Dans un deuxième temps, nous avons étendu le modèle proposé pour permettre un déploiement propre et efficace de politiques de contrôle d'accès complexes contenant des exceptions et/ou des règles de conflit sur les services NFV. Les résultats expérimentaux montrent que, par rapport aux solutions similaires existantes, l'extension proposée réduit de manière significative (1) la complexité de la spécification DTE à déployer et (2) l'impact sur les performances du réseau du service NFV dans lequel les politiques sont déployées.

Dans un troisième temps et en considérant un modèle d'adversaire fort dans lequel un adversaire interne contrôle un ou plusieurs nœuds inconnus (VNF) qui composent le service NFV, nous proposons un nouveau modèle de déploiement des politiques de contrôle d'accès correctement prouvable qui calcule de manière optimale le bon ensemble de règles de la politique de contrôle d'accès qui doit être déployé dans chaque lien réseau reliant deux VNFs, rendant impossible la violation de l'exigence de contrôle d'accès appliquée par l'adversaire interne.

Dans un quatrième temps, visant à trouver le meilleur compromis entre les ressources nécessaires au déploiement de la politique de contrôle d'accès et l'impact sur la qualité du service NFV dû au déploiement des exigences de contrôle d'accès, nous proposons un modèle permettant de quantifier l'impact en terme de latence du déploiement du modèle de contrôle d'accès sur le service NFV cible. Ensuite, nous définissons un problème de minimisation qui vise à réduire à la fois l'impact en termes de latence dans un service NFV et les ressources de calcul et de stockage qui sont requises par les points de déploiement de la politique de contrôle d'accès. Par la

suite, nous montrons que le problème d'optimisation proposé est non linéaire et non convexe et nous utilisons une amélioration de l'algorithme génétique de tri non dominé NSGA II pour le résoudre. Enfin, nous construisons un cadre de simulation pour évaluer l'efficacité du modèle d'optimisation proposé. Les simulations réalisées montrent que le modèle d'optimisation que nous proposons permet aux administrateurs de la sécurité d'avoir des informations claires et suffisantes pour prendre des décisions sur le bon compromis entre la dégradation de la qualité de service et la quantité de ressources qui devraient être consacrées aux points de déploiement de la politique de contrôle d'accès.

REMERCIEMENTS

La réalisation de cette thèse n'a pas été possible sans le soutien, l'aide, l'orientation et les conseils que j'ai reçus de nombreuses personnes au cours de ma vie d'étudiant. Je tiens sincèrement à exprimer ma gratitude à toutes ces personnes, du fond du cœur.

Je tiens à exprimer ma gratitude à mes directeurs de thèse ***David Espes*** et ***Marc-Oliver Pahl*** pour leur soutien, leur confiance et leurs conseils qui m'ont permis d'atteindre cet objectif important. Je tiens à remercier ***Nora Cuppens-Boulahia*** et ***Frédéric Cuppens*** pour leur support et leur confiance tout au long de la première année de ma thèse. Ce fut un honneur pour moi d'être l'un de leurs doctorants. Je tiens à exprimer ma gratitude à ***David*** pour ses précieux conseils et pour ses réponses à toutes mes questions tout au long des trois années de ma thèse.

Je tiens également à remercier ***Joaquin Garcia-Alfaro*** et ***Rémi Badonnel*** qui ont eu la lourde tâche de rapporter ma thèse et de donner leurs conseils pour améliorer son contenu. Un grand merci à ***Abdelmalek Benzekri*** et ***Françoise Sailhan*** pour avoir fait partie du jury de ma thèse.

Un grand merci à tous mes amis et collègues, membres de notre équipe pour avoir apporté de la joie dans ma vie. A tous ceux qui m'ont soutenu durant cette thèse, merci encore.

Je tiens à remercier tout particulièrement mon bien-aimé mari ***Anis Bkakria*** pour son soutien continu. Je ne peux pas te remercier assez de m'avoir encouragée tout au long de cette expérience. Tu as été ma source d'inspiration. Je n'aurais jamais pu réaliser cette thèse sans ton amour, ton soutien, tes conseils et tes compréhensions. Merci pour tout ce que tu as apporté à ma vie. Merci de m'avoir donné deux merveilleux enfants ***Iyed*** et ***Mirna***. J'ai de la chance de t'avoir rencontré et j'ai hâte de passer le reste de ma vie avec toi.

Un grand merci à ma famille. Avant tout, ma mère ***Raja*** et mon père ***Abdelwahab*** pour avoir été ma motivation à réussir dans la vie. Je tiens également à remercier ma sœur ***Marwa***, mes frères ***Safwen*** et ***Mahdi***, à mes ***oncles*** et ***tantes***. Je tiens également à remercier ***ma belle-famille*** pour leur compréhension et leurs encouragements.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Contents

1.1 Context and Motivation

To respond quickly and efficiently to the growing need for new and diverse network services, today's networks must be flexible, adaptable, responsive, robust and highly resourced (processing capacity, storage and bandwidth) [MS14]. The model according to which the networks of previous generations are built, which relies on proprietary hardware often strongly coupled to a particular technology (IPv4 for example), is unable to respond effectively and quickly to these needs to achieve this transformation. It is therefore from this observation that the Network Function Virtualisation (NFV) paradigm was born. With NFV, network functions such as storage, computation, filtering, and NAT, can be decoupled from the hardware, which allows them to be virtualized in Virtual Network Function (VNF) to be installed anywhere on the network infrastructure both physically and virtually. The goal is then to simplify network service lifecycle management, to optimize the resources and improve operational efficiency and to allow rapid development of new network services while maximizing flexibility for scalability and automation.

With the NFV paradigm, enterprises are tempted to virtualize and outsource their network infrastructures to the cloud, in order to reap the benefits of the latter. For cloud providers, NFV opens the door to new highly innovative and lucrative business models such as Network as a Service (NaaS) [LRMO16]. Large number of cloud providers

already offer a broad spectrum of out-of-the-box VNFs e.g., load balancer [Gui18], AWS Transit Gateway [AWS], and AWS Network Firewall. The companies can then quickly create efficient network services by linking various VNFs. According to a study conducted by Meticulous Research [Ltd20], The global NFV market is expected to grow 34.9% annually to reach 122 billion by 2027.

Despite the unanimous recognition of the benefits of outsourcing network functions to the cloud through NFV, the security remains to be one of the vital concerns and potential hurdles that prevent wide-spread adoption of NFV [PHS+18, ZAR22]. NFV significantly expands the attack surface as it relies on a broad software stack such as hypervisors (e.g., KVM, Docker), automation and management tools (e.g., OpenStack, Cloudify, Ansible), and network isolation and data plane acceleration software (e.g., OpenVswitch, FD.io). Each of these software products may expose various vulnerabilities (e.g., CVE-2020-35498 and CVE-2020-3236), increasing tenfold the opportunities for an adversary to violate enterprise network service specifications [ZAR22]. Moreover, in NFV platforms, VNFs belonging to different tenants (enterprises) often share the same physical or virtual server. A malicious tenant can then conduct side-channel attacks [AQK+19,AWJ20] against other tenants' VNFs to steal or corrupt sensitive data, or disrupt the different features provided by the VNFs. It has been shown by Youngjoo Shin et al. [SKH20] that shared access to the CPU cache can allow one tenant to infer information such as the firewall filtering rules that are enforced in another tenant's network service.

Companies' and Organizations' lack of confidence in NFV environments inhibits the outsourcing of network functions to the cloud. This critical gap must be bridged by the NFV research field which needs to be fed by new security-oriented approaches that can be used to improve the security posture of existing NFV environments.

1.2 Research Goal and Questions

The goal of this dissertation is to improve the security of NFV services. To meet the previously stated objective, several research questions need to be answered.

First, considering the fact that VNFs are software-based network functions running on virtualized infrastructures, the security threats related to NFV based network services can be larger than traditional network services, ranging from generic virtualization threats [AQK+19], physical network functions prior to virtualization [BMMR+15],

and threats introduced by the combination of virtualization technology with networking [AWJ20] such as orchestration and policy violation. Therefore, the first sub-goal is then to establish a comprehensive NFV layer-specific threat taxonomy by answering the following research questions:

- **RQ-1**: What are the security threats in NFV infrastructures?
- **RQ-2**: Which layers of the NFV infrastructure are concerned by which security threats?

To overcome some of the threats and vulnerabilities related to NFV infrastructures, several security mechanisms have been proposed in the literature. One of these mechanisms is the enforcement of access control policies. Hence, as a second sub-goal, we focus on studying existing mechanisms that can be used to enforce access control policies on NFV services. Then, to evaluate and compare existing access control policy enforcement in NFV services solutions, as a third sub-goal, we focus on identifying the set of relevant properties that need to be satisfied to effectively enhance the security of NFV services. We fulfill the previous two sub-goals by answering the following research questions:

- **RQ-3**: What is the state of the art with respect to access control policy enforcement in virtualized network infrastructures?
- **RQ-4**: What are the relevant properties that need to be satisfied by access control enforcement mechanisms to enhance the security of NFv services in existing solutions?

By answering the previous questions, we observe that all existing access control enforcement on virtualized infrastructure suffers from at least one of the following limitations. First, none of the existing approaches provides formal proofs supporting the correctness of the deployment of access control policies. Second, most of the existing approaches rely on specification models that are not expressive enough to handle high level policies specified using other access control models such as RBAC [SCFY96], ORBAC [KBB$^+$03], and ABAC [PFMP04]. Finally, several of the existing approaches require the modification and the management of the NFV Infrastructure (NFVI) which makes them non-compliant with the ETSI-NFV architecture and thus not easily used

in practice. Therefore, to overcome the previous limitation, as a forth sub-goal, we focus on the definition of a provably correct, highly expressive, and ETSI-NFV compliant access control enforcement mechanism by investigating the following research questions:

- **RQ-5.1**: How to define a sufficiently expressive high-level specification model that allows to correctly express requirements modeled using well known access control models?
- **RQ-5.2**: How to correctly refine the high level access control policy towards concrete-level deployable requirements?
- **RQ-5.3**: How to define an ETSI-NFV compliant deployment of the concrete-level requirements?

The investigations we conduct to answer the previous research questions allow us to observe that high-level access control policies are often composed of a set of authorizations and prohibitions, which may give place to conflicts and exceptions in the policy to be enforced. Consequently, as a fifth sub-goal, we focus on the definition of a solution allowing efficient enforcement of high-level mixed access control policies (i.e., policies that contain both positive and negative authorizations) containing exceptions in virtualized infrastructure. To meet the previous sub-goal, we investigate the following research questions:

- **RQ-5.4**: Can the solutions proposed to answer RQ-5.1, RQ-5.2, and RQ-5.3 be used to efficiently (i.e., by reducing to the best the number of rules in the concrete-level policy as well as the time needed to evaluate an access request) deploy high-level mixed access control policies containing exceptions?
- **RQ-5.5**: How can we improve the proposed solutions to deal efficiently with high-level mixed access control policies containing exceptions?

NFV services, as any other virtualized infrastructure, can be compromised and partially controlled by an adversary. The conducted literature review (Chapter 2) shows that all existing access control policies enforcement solutions consider only outsider adversaries that aim to remotely bypass the enforced policy. Hence, the sixth sub-goal we consider in this dissertation focuses on proposing a solution that can correctly enforce

access control policies when both outsider and insider adversaries are considered. To meet the previous sub-goal, we investigate the following research question:

- **RQ-5.6**: How to deal with both insider and outsider adversaries? More specifically, how to correctly deploy access control policies when an unknown part of the target NFV service is compromised and controlled by an adversary?

Finally, the enforcement of access control policy requires resources in terms of computation and storage, and often impacts the functionality provided by the target NFV service i.e., by introducing latency due to the traffic analysis and rule enforcement. The conducted literature review (Chapter 2) allows us to observe that none of the existing access control enforcement on virtualized infrastructure solutions has considered minimizing both resource consumption and the impact related to access control policy enforcement. Hence, the seventh sub-goal of this dissertation focuses on proposing an access control policy enforcement solution that allows minimizing both the resources needed by the policy enforcement points and the impact in terms of latency introduced by the access control policy deployment. We achieve the previous sub-goal by investigating the following research question:

- **RQ-5.7**: How to define an access control policy enforcement mechanism that minimizes both the consumed resources and the impact on the target NFV service caused by the access control policy deployment?

1.3 Methodologies and Contributions

To meet the above research goals and answer the defined aforementioned research questions, this dissertation provides the following contributions.

- An in-depth analysis about NFV security and existing access control policy enforcement solutions for virtualized infrastructures. To answer the research questions RQ-1 and RQ-2, in this contribution, we first conduct a literature review of existing and potential security issues and threats in the NFV architecture. Then, we classify them according to the components of the NFV architecture that are concerned by these threats. At a second time, to answer the research questions RQ-3 and RQ-4, we review existing solutions for enforcing access control policies

in NFV infrastructures. Then, we identify the relevant properties that need to be satisfied by access control enforcement mechanisms to effectively enhance the security of NFV services. Third, we give an extensive comparative overview of the existing access control enforcement solutions regarding the identified properties. Finally, we identify several open challenges that we addressed throughout the following contributions.

- A provably correct, highly expressive, and ETSI-NFV compliant access control enforcement model. Specifically, to answer the research question RQ-5.1, we propose a formal model that provides a software-defined access control as a service capability for network services. First, we define an expressive specification model to be able to express high-level access control requirements to be enforced over virtualized network services. Then, we show that our specification model can correctly express access control requirements specified using well-known existing access control models such as RBAC [SCFY96], ORBAC [KBB+03], and ABAC [PFMP04]. Second, to tackle RQ-5.2, we propose a provably correct method for refining high-level access control requirements towards a Domain Type Enforcement (DTE) concrete specification. Finally, to answer RQ-5.3, we define an ETSI-NFV compliant, efficient, and scalable access control policy enforcement method, as illustrated by the conducted experimental evaluations.

- A priority-based DTE for exception management. In this contribution, to answer RQ-5.4 and RQ-5.5, we define a solution allowing a clean and efficient deployment of complex access control policies containing exceptions and/or conflict rules on NFV services. Our model allows a clean deployment in the sense that, compared to the high level security policy to be deployed, it does not introduce additional low-level rules which allows security administrators to straightforwardly understand and update the deployed concrete-level policy. Our model relies on a provably correct approach for exception management in DTE specification. The conducted empirical evaluations show that the priority-based DTE model we are proposing is quite efficient for enforcing big and complex policies that contain exceptions.

- An approach allowing an optimal access control deployment in NFV services. In this contribution, we consider a strong adversary model by assuming that we are dealing with an insider adversary who can control one or more unknown nodes

(VNFs) that compose the NFV service. Hence, to answer RQ-5.6, we propose a new correctly provable access control policy enforcement model that optimally computes the right set of rules of the access control policy that needs to be deployed in each network link linking two VNFs, making violating the enforced access control requirement not possible for the insider adversary. As a second time, to tackle RQ-5.7, we propose a formal modeling of the optimal deployment of the access control policy problem allowing to model and quantify consumed resources as well as the impact in terms of latency that is to be generated by the access control policy deployment. We show that the optimal deployment of access control policies problem is a nonlinear multi-objective optimization problem and uses an improvement of the Non-dominated Sorting Genetic Algorithm NSGA II [WSZ+18] for solving it. Finally, we conduct an experiment in an emulated Internet environment, NS-3 [ns3], to evaluate the effectiveness of access control policy deployment solutions.

1.4 Outline of dissertation

The organization of this dissertation is given as follows:

- **Chapter 2 – Access Control in NFV: State of the Art & Background** – gives a review of existing and potential security issues and threats in the NFV architecture. It classifies them according to the components that are concerned by these threats. Then, it reviews existing solutions for enforcing access control policies in NFV infrastructures.
- **Chapter 3 – A Domain Type Enforcement of Access Control Policies in NFV Services** – defines an expressive specification model of high-level access control requirements to be enforced over network services. Then, it proposes a provably correct method for refining them towards a Domain Type Enforcement (DTE) concrete specification.
- **Chapter 4 – A Priority-based DTE for Exception Management** – investigates the management of access control exception in concrete-level DTE specification. Then, it proposes a solution to allow a clean and efficient deployment of complex access control policies containing exceptions and/or conflict rules on NFV services.

- **Chapter 5 – Optimal Access Control Deployment in NFV Service** – proposes a formal modeling of the optimal deployment of the access control policy problem allowing to quantify and optimize consumed resources as well as the impact in terms of latency caused by the access control policy deployment.
- **Chapter 6 – Conclusions and Perspectives** – concludes the thesis with a summary of the contributions and presents the perspectives of future works.

CHAPTER 2

ACCESS CONTROL IN NFV: STATE OF THE ART & BACKGROUND

Contents

2.1 Introduction

Network Function Virtualization (NFV) aims to make the physical equipment used versatile by allowing them to multiply the functions they can perform (each function becoming a software rather than a physical device). NFV increases the flexibility and cost-effectiveness of network services by eliminating the bottlenecks imposed by manual processes, while making it possible to deploy new services on demand [TKJ16, HGJL15]. With NFV, providers are able to deliver services faster, lower their cost, and leverage automation to match customer requirements for scalability and agility. NFV extracts network functions, allowing software running on standardized compute nodes to install, control, and manipulate them. Despite the aforementioned advantages, existing NFV infrastructures are suffering from several threats and vulnerabilities [RAB+16] such as security breaches, data loss and information leakage, malicious insiders, etc. To mitigate these threats, several security mechanisms have been defined in the literature such as Intrusion Detection and Prevention [AZZ+20], Access control and Flow Control Policy [Jae15, PHMZ16, SECBC20, TCCM17], and Data Encryption, etc.

As it was mentioned in Chapter 1, the research goal of this dissertation is to improve the security of NFV services through the use of access control policy enforcement techniques. To meet the previous objective, we first need to establish a comprehensive NFV layer-specific threat taxonomy by answering the following two research questions:

- **RQ-1:** What are the security threats in NFV infrastructure?
- **RQ-2:** What are the security threats that can be addressed/reduced by Access control policy enforcement?

Then, we focused on studying existing mechanisms that can be used to enforce access control policies on NFV services to understand their strengths and limitations. To meet the previous objective, we investigate the following two research questions:

- **RQ-3:** What is the state of the art with respect to the access control policy enforcement in virtualized network infrastructure?
- **RQ-4:** What are the relevant properties that are satisfied/unsatisfied in existing solutions?

The contribution of this chapter is threefold. First we give a review of existing and potential security issues and threats in the NFV architecture. Then, to allow a better

understanding of the impact of these threats, we classify them according to the components that are concerned by these threats. Second, we review existing solutions for enforcing access control policies in NFV infrastructures. Then, we give an extensive comparative overview of the existing solutions regarding several properties including, the considered adversary model, the effectiveness, the correctness, the optimality, etc. The comparative overview illustrates the main improvement we bring through the contributions we propose in this thesis compared to existing solutions.

This chapter is organized as follows. Section 2.2 introduces the NFV architecture. Section 2.3 study the security in NFV by proposing a classification of the threats related to the NFV architecture components. Section 2.4 study the security contermeaures. Section 2.5 propose a clasification of enforcement security policies approaches over NFV. Finally, section 2.7 concludes this chapter.

2.2 NFV architecture

The NFV architecture proposed by the European Telecommunications Standards Institute (ETSI) helps to define the standards for implementing network function virtualization. Each component of the architecture is based on these standards to provide a higher level of stability and interoperability. This section introduces the main architectural components of NFV that is provided by ETSI in [E+14]. The NFV architecture includes the three following main functional blocks: NFV Infrastructure (NFVI), Virtual Network Functions (VNFs), and NFV Management and Orchestration (NFV MANO) interconnected over specific reference points, as illustrated in Figure 2.1.

2.2.1 NFV Infrastructure (NFVI)

NFVI is a low-cost based infrastructure that contains a set of hardware and software components (compute, storage, network), which supports the software and delivers the environment on which virtualized network functions (VNFs) are deployed, managed, and executed. NFVI provides the virtualization layer to ensure the separation of software from hardware for network functions such as a hypervisor like KVM, or a container management platform. as well as the physical compute, storage, and network components that host the VNFs. So that they can be logically partitioned and provisioned to support the VNFs. The NFVI module is also essential for building complex, widely

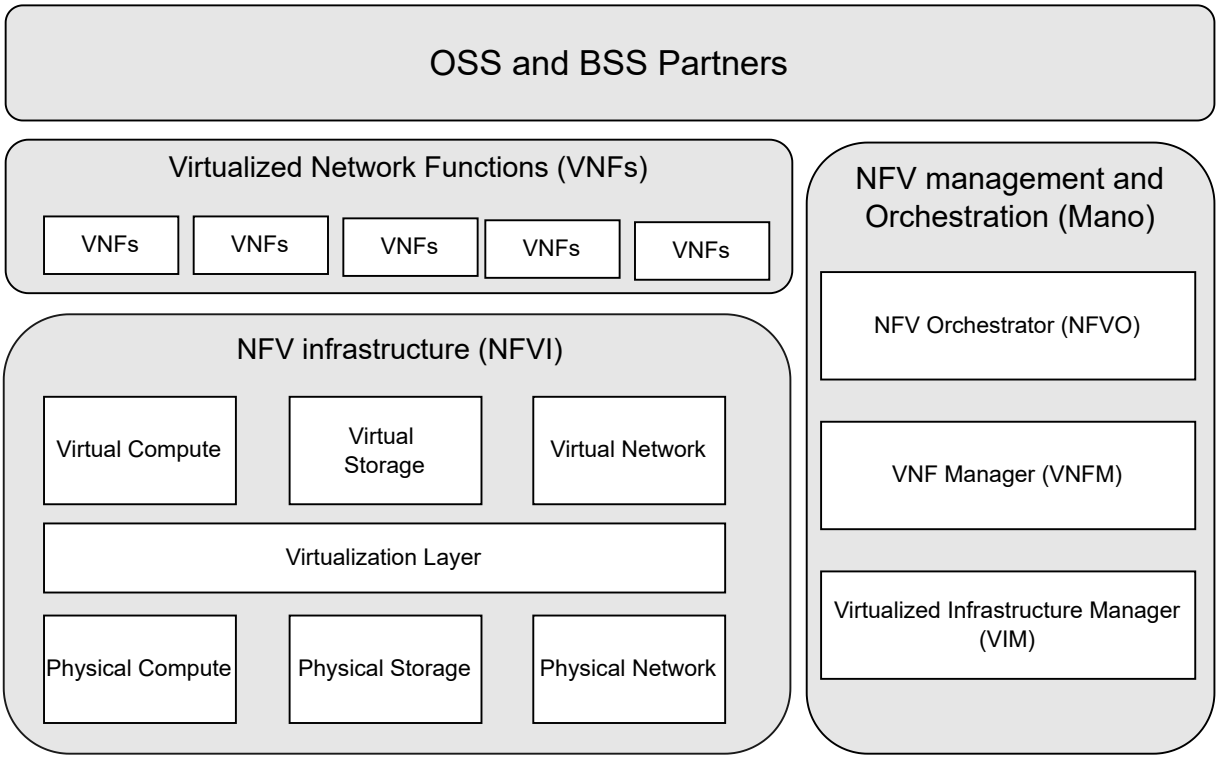


Figure 2.1 – NFV infrastructure

distributed networks without the geographic limitations associated with the traditional network architectures. The NFVI module is administered by the NFVI Infrastructure Manager (VIM).

2.2.2 Virtual Network Functions (VNFs)

VNFs are software packages where the implementation of network functions takes place using software based approaches. A number of VNFs that are executed in one or more virtual machines on top of the hardware networking infrastructure can be chained to provide virtual network services. VNFs comprise routers, switches, firewalls and a large number of other network services now offered as software from vendors such as Cisco and Juniper Networks. VNFs use the virtualized infrastructure provided by the NFVI module to connect to the network and provide programmable and scalable services. VNF managers support the lifecycle of VNF instances and the management of VNF software. The virtualization of network functions reduces the implementation cost and the use of the material and improves the scalability [LL21]. Using a network func-

tions virtualization architecture, VNFs are deployed on demand, which eliminate the deployment time associated with traditional network hardware, as well as the requirement for on-site technical expertise when deploying remotely. VNFs offer the agility to meet or anticipate dynamic network performance or expansion demands in hybrid and multi-cloud environments [ZPL+20].

2.2.3 NFV Management and Orchestration (NFV MANO)

NFV MANO is a framework developed by a working group of the European Telecommunications Standards Institute (ETSI) to provide the overall management and orchestration process for the resources - NFVIs and VNFs - that operate in a virtualized data center including computers, networks, storage, and virtual machines (VMs) [Ers13]. The MANO component executes network services through automation, provisioning, and workflow coordination on VIM and VNF managers that perform VNF functions and overlay networking service chains. NFV MANO uses templates for standard VNFs enabling architects to select the suitable NFVI resources to deploy. MANO connects the NFV architecture to existing OSS/BSS systems and is composed of three components:

- Virtualized Infrastructure Manager(s) (VIM): The VIM is in charge of managing the used resources of the NFVI such as compute, storage, and network resources. It monitors the lifecycle of virtual resources allocated to a domain, keeps the virtual machine and physical machine matched, analyzes hardware, software and virtual performance through an agent, and collects infrastructure failure information. An example of an open source VIM manager is OpenStack that controls physical and virtual resources. The Red Hat OpenStack platform is an example of a commercial VIM.

- NFV Orchestrator (NFVO): The NFVO plays a very important role in managing resource orchestration. It coordinates the allocation of hardware resources: authorizes, scales, releases physical resources across the set of DataCenters. It gives orders to the hardware resource manager VIM. Additionally, the NFVO handles service orchestration by checking the establishment or the release of one or more VNF virtual functions. The NFVO relies on resource catalogs that specify the desired template to handle network services such as:

  - VNFD catalog includes the descriptor of each instance of VNF

  - Service catalog allows to enumerate all the VNF functions to be chained to obtain a sub-network of instances
  - NFVI catalog contains the resources required to implement an NFV service.

- VNF Manager(s) (VNFM): manages the lifecycle of virtual network functions such as creation, scaling, deployment, configuration. It is able to release their instances, monitors and detects their faults. It provides a coordinated and adaptive role for NFVI and element/network management system configuration and event reporting. The VNFM can be deployed for each VNF or multiple VNFs.

Established network vendors like Cisco and Juniper provide the functionality of NFV MANO, as well as open-source offerings from Cloudify and Open Source MANO.

2.3 Security in NFV

NFV technologies are increasingly being utilized by e-communications operators to answer rapidly to the market needs. It consists of the virtualization of network functions, which is most often telecommunication network applications including routers, firewalls, load balancers, etc. The main goal is to allow these functions to run as software programs or virtual network functions (VNFs). Since service providers do not exhibit internal structures, users can have limited ability to see and control network resources [LV94]. These limitations inherently lead to important security threats [DSVV17, LROT17, RLOH17, SECBC20, SECB+20]. According to what we have presented in Section 2.2, the NFV architecture is composed of several components (VNFs, NFVI, and NFV MANO). Each of these components may introduce different threats. In this section, we propose a classification of the threats related to the NFV architecture components.

2.3.1 Security issues related to NFVI

In this section, we discuss the set of threats related to the different layers that compose the NFVI. Figure B.1 gives a taxonomy of existing threats according to the impacted NFVI layer.

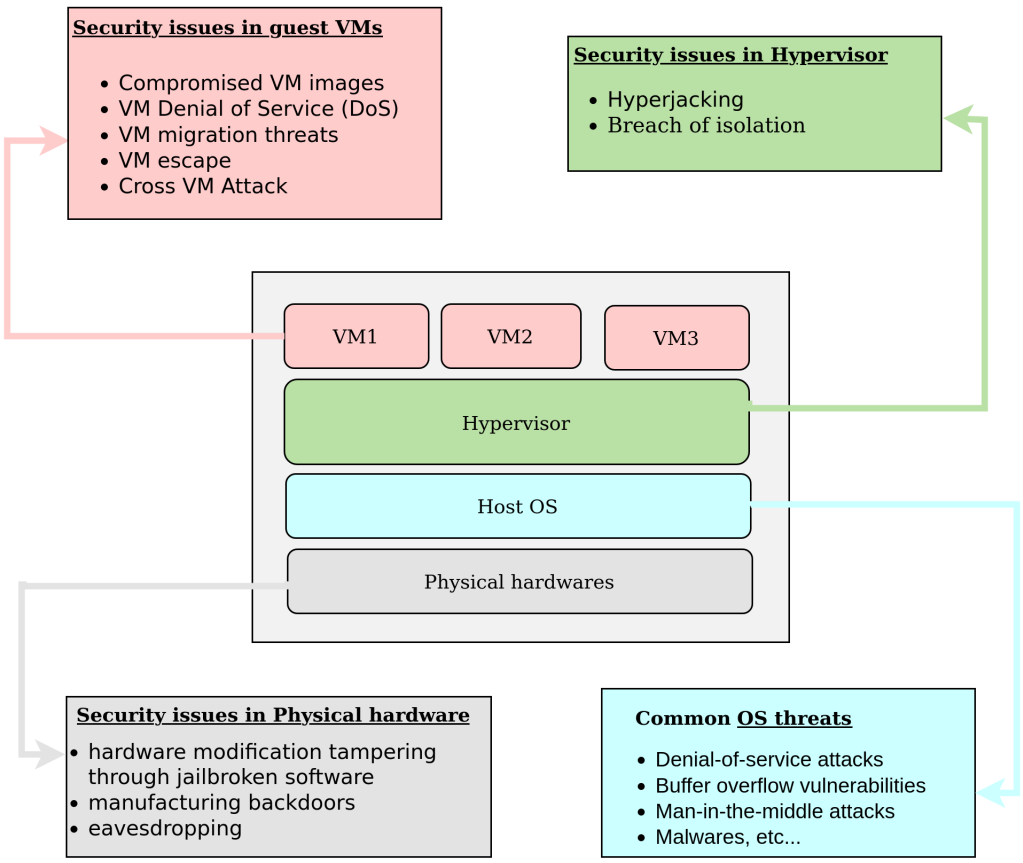


Figure 2.2 – NFVI threats

Security issues related to virtual machines

- **Compromised VM images:** To launch VM instances on demand, a VM image is used as a software template pre-packaged containing a configuration files. Network operators can create their personal VM images from scratch or download VM images that are stored in the public repository of provider. In some cases, users can download or upload a VM image. If a malicious user upload a VM images containing malicious code and if another user runs this image, all their virtual machines will be infected with the hidden malware [EMEBHM16, ABET16]. This kind of threat represents a serious threat in the virtualized environment [AH21].
- **VM Denial of Service (DoS):** Using the virtualization, the physical resources like memory disk, CPU, and network bandwidth are shared by several virtual machines. The DoS attack can be launched by an attacker that tries to exploit a misconfiguration of the hypervisor [PBB13] by using all the resources that are

available [HRFMF13, HCSL12].

- **VM migration threats:** Virtual machine (VM) migration is the main feature of cloud computing that is mainly used for maximum availability, workload balancing, maintenance of hardware and catastrophe recovery. It permits network operators to transfer virtual machines from one physical machine to another [KSK14]. This beneficial feature could be vulnerable to several types of attacks like traffic sniffing, DDoS flooding, and man-in-the-middle attacks [DTM10, PBB13, FJKK17]. For the secure migration of VMs, a lot of research has been done focusing on offline migration [CGS+17]. Particularly, it has been shown that offline migration can be performed securely. However, live VM migration has yet to be actively researched. Live VM migration has multiple vulnerabilities and threats [SM14, YMH+18, GXWS20] that can be easily exploited by the attackers. Due to their demonstrations, the live VM migration could affect any of these three various classes: (1) control plane, (2) data plane and (3) migration module.

  - Control plane; The whole network services can be impacted by any vulnerability or attack [G+16, ZCP+15] occurring at the control plane level. Multiple security issues in the control plane can allow an attacker to exploit the live migration operation. Mainly, an attacker can perform the following attacks:
    - Attack on VM monitor (VMM) and VM [SLCL09]: A virtual machine having malicious code can be migrated by an attacker to a host server containing the compromised VM. This malicious code is used to exchange information between the VMM (that virtualizes the machine's resources in terms of CPU, memory, storage, network, and I/O) and the target VM via a hidden channel. This channel will make the host server's privacy compromised by divulging the target VM's information.
    - Denial-of-Service (DoS) attack [Cho13]: A multiple virtual machines will be created by the adversary on the host operating system in an attempt to overload the host OS and making it incapable to accept additional migrated virtual machines.
  - Data plane: In the data plane, many contents of memory such as application data and kernel states are moved from source to destination server. This transfer of data is migrated as a plaintext data without any encryption. Multiple passive and active attacks are able to be launched on it [BR+11].

    * Active attacks: Among the most severe attack is the active one in which there is a memory exploitation by the attacker while the VMs are being live migrated such as alteration in explicit memory zone of advent VM, authentication service and Network storage attacks [AKS13].
    * Passive attacks: The communication channel and other network flows are observed by the attacker in order to obtain information from the migrating VM [ASS13]. So, the attacker can get the authentic guest by obtaining the VM properties such as the duration and the size of the migration and some information of the migrated VM such as application data capturing packets, keys, and passwords.

  – Migration module: is a software of VMM for migration process which can be used for live migration of virtual machines. Both guest operating system and host system are able to interact with each other. In addition, the host system has a total control over every virtual machine that runs on its VMM. If the VMM is compromised by the attacker through its migration module, the integrity of all guest VMs running on top of this VMM will be compromised. So, all virtual machines that migrate to the affected VMM in the future will also be compromised [HKKT10, SAS12, CSS+12]. This plane could be vulnerable to several types of attacks such as DDoS flooding attack and man-in-the-middle attack [DTM10, FJKK17]

- **VM escape:** This attack concerns malicious virtual machines that manage to break the spatial isolation set up by the virtual machine manager. The attacker can then access the memory of other machines, physical devices, and the hypervisor itself. If successfully performed, VM escape can allow an attacker to monitor other VMs, including accessing their shared resources and CPU utilization [GWS10].

- **Cross VM Attack:** Known also as VM hopping attack is a process allowing an attacker who is controlling a VM to maliciously gain access to another VM running on the same physical host by exploiting weaknesses in virtual infrastructure or in the hypervisor [XX12]. VM hopping is shown by Ristenpart et al [RTSS09] to be a quite realistic attack.

Security issue related to Hypervisors: The isolation of virtual machines is provided by the hypervisor that plays a key role. This latter provides services and protection for guest VMs. It often has a significant attack surface by an attacker who is residing in one VM and then threatens the security of other co-located VMs. That is why all the attached VMs to an infected hypervisor could also be infected. We present the two possibilities to make an hypervisor compromised as follows:

- **Breach of isolation:** Conceptually, hypervisor-based isolation ensures that (1) the content i.e., data and processes of each VM must be isolated from others, and (2) each VM will use only the granted resources. Bad configuration and/or weaknesses in the hypervisor may give rise to a possible VM escape or DoS attacks [SAD+12, LTD17].

- **Hyperjacking:** This attack allows a malicous adversary to take the control of the hypervisor to create the virtual environment within a VM host [IMZ16]. This may allow the adversary to control the virtualization layer and thus taking control of all the VM that are running on top of this layer [MA17].

Security issue related to Host OS: In NFV infrastructure, virtual machines are running on top of the host operation system. Hence, if the latter get compromised by an adversary that may very likely be able to compromise all the NFVI layers running on top of the host operating system including the hypervisor and the guest virtual machines. Host OS are threaten by the common operating system threats, such as malwares, network intrusion, buffer overflow, denial of service, etc.

Security issue related to Physical hardwares: To have a secure system, it is very important that the developer find better methods of designing and exhaustive testing of his system to be sure that it will not be broken and redesigned in a few weeks. The reason is that once the hardware module is affected by the attackers, the security mechanisms of the software on these devices will also be affected. In the literature, several kinds of hardware attacks are reported in [Pag13, BMMR+15] such as hardware modification tampering through jailbroken software and manufacturing backdoors eavesdropping.

2.3.2 Security issues related to VNF

In this section, we will present the set of threats in the VNF layer:

- **DoS attack:** The NFV environments can have a major threat such as Denial of Service attacks (DoS). DoS is one of the most popular and destructive types of cyberattacks. This latter generates a large number of anomalous packets from a compromised VNF and directly target other VNFs that are running on the same hypervisor or even on different hypervisor. As a result, this attack overloads the buffer of the network devices, generates amplified traffic to occupy the bandwidth between the data and the control plane, and consumes all available resources (CPU, memory) in a short time in order to make them inaccessible [JSGI09]. A huge volume of traffic from a compromised VNF can be generated and sent to other VNFs that would be running on the same hypervisor or even on different hypervisors. Similarly, some VNF applications can consume high CPU, hard disk, and memory resources in order to exhaust the hypervisor

- **Vulnerabilities in VNF softwares:** The network operators use VNF softwares to launch and deploy network functions on request. Unfortunately, different kinds of software flaws provide opportunities for security issues that threaten network functions when deployed in an NFV environment [RAB+16] like taking advantage of a buffer overflow to execute a random code.

2.3.3 Security issues related to NFV MANO

- **Failure of troubleshooting:** When a failure occurs in the NFV architecture, the NFV management and orchestration entity need to gather, analyse, and correlate the logs generated by several miscellaneous entities to determine the root causes of the failure. The heterogeneity of the entities and services composing the NFV infrastructure makes failure detection and troubleshooting challenging. LAl et al [LTD17] shown that it is even more challenging when compromised VNFs maliciously generate huge amount of logs making analysing other VNFs logs extremely difficult.

- **Attacks to management and control plane:** All operations, that comprise the configuration, creation, management, and other functionalities of NFV network

services are controlled by the NFV management and control plane. Nevertheless, this NFV management and control plane can be rendered a unique failure point. As a result, the compromising of any control operation may often lead to a failure of either specific network functionalities or the global virtualized system [ZCP+15].

2.3.4 Common security issues for the three components:

- **Malicious insiders:** An insider is a person within an organization who has authorized access, privilege, or knowledge of information systems, information services, and missions. He is a person motivated to adversely impact an organization's mission through a series of actions that compromise the confidentiality, integrity and/or availability of information [LTD17].

- **Insecure interfaces:** The security of a virtualized system depends also on the security of the interfaces. The latter depends on the type of interface, manufacturers, etc. A study conducted by the European Union Agency for Cybersecurity (ENISA) [ENI22] reports that insecure interfaces such as TLSv1, FTP, SNMP are still used in NFV services. These interfaces can be easily compromised by exploiting vulnerabilities in the used protocols such as spoofing and insecure settings. The attackers also may integrate malicious code in the interfaces in order to obtain an unauthorized access to the target's system [WBO15, LTD17].

- **Security policy and regular compliance failure:** Generally, the connectivity between VNFs is controlled by the management and orchestration module. This latter can build relations between virtual infrastructure resources and VNFs. Nevertheless, the inconsistency in the way network services are orchestrated, maintained, and used can lead to security risks [YF16, G+16].

- **Data loss and information leakage:** The loss or leakage of data represents a major problem that needs to be solved in all kinds of organizations. Ristenpart et al. [DSPA18] have shown that a virtual instance (NFV or VM) controlled by a malicious adversary can be used to learn and leak information about co-resident virtual instances through the exploitation of specific side channels. In particular, the authors of [DSPA18] have shown that the time-shared caches can be used by

an attacker to figure out when other virtual instances are experiencing computational load.

2.4 Security Countermeasures

Several security mechanisms have been proposed in the literature to mitigate the aforementioned threats. Among the security mechanisms that can overcome the above security problems in the different NFV layers, the enforcement of access control policies. As reported in [Pat19], once well specified and correctly enforced, access control policies can help reducing several security threats such as, DoS attack, data leakage, breach of isolation, cross VM Attack, malicious insiders, etc. The following table list the set of security threats that can be addressed by access control policy enforcement.

Security Threats	**Can be addressed ?**
Security issues in guest VMs	✓
Security issues in hypervisor	
Insecure management interfaces	
Compromising virtual network components	✓
Malicious insiders	✓
Untrustworthy service composition	✓
Hardware attacks	
Vulnerabilities in VNF softwares	✓
DoS/DDoS attacks	✓
Attack to management and control plane	✓
information leakage	✓

Table 2.1 – The virtualized system threats and vulnerabilities that can be addressed by access control policy enforcement [Pat19]

According to the previous table, most existing threats to virtualized systems can be addressed by access control policy enforcement. Hence, in this thesis we focused on the study of access control policies enforcement in NFV services.

2.5 Access Control in NFV

As we have seen in Section 2.3, NFV environments bring new security challenges and issues introduced by several types of threats and vulnerabilities [DSVV17, LROT17,

RLOH17, SECBC20, SECB+20]. In recent years, several researchers investigated the usage of policy-driven security mechanisms, such as access control, to mitigate threats raised by malicious user-related vulnerabilities (e.g., unauthorized access and/or privilege, unauthorized flow between VNFs). Thus, many approaches have been proposed to define and enforce security policies over NFV architecture to ensure their security. These approaches can be classified into two classes: SDN-based policy enforcement and Orchestrator-based policy enforcement.

2.5.1 SDN-based Access Control

Nowadays, the majority of the existing frameworks are mainly oriented into migrating network functions from hardware devices to a virtualization environment. SDN and NFV are two independent but complementary concepts. Their combination offers a definite added value in terms of automation, service mobility, operation cost reduction, and security management. Following the previous idea, some approaches have used SDN to enable access control policy enforcement solutions over NFV infrastructure.

Yakasai and Guy [YG15] propose a virtualized network access control function using OpenFlow protocol [MAB+08]. The proposed solution enables network access control in SDN architectures by combining an implementation of 802.1X framework with an authorization method through a stateful role-based firewall enforcement. The solution allows a high-level specification of the access control policy to be deployed based on the role that each network function/component will play in the NFV service. The update and deployment of the defined security policies can be made dynamically and instantaneously enforced. Nevertheless, the proposed approach is deeply dependent on the SDN architecture which is required to be installed and configured in the considered NFV infrastructure. Hence, the usage of this solution within an NFV architecture requires the control and/or the management of the NFV infrastructure. Therefore, it cannot be directly used in NFV as service architecture, but only in NFVI as a service architecture. Moreover, the proposed solution relies on a single access control policy enforcement point which can be a bottleneck when dealing with large traffic NFV service. The latter drawback can be addressed by considering several policy enforcement points. However, the authors did not give details on how the proposed solution can be extended to a multi policy enforcement points approach.

To address the single policy enforcement point limitation described above, Leopoldo,

et al. propose ACLFLOW [MRD18], an approach that combines NFV and SDN frameworks to provide a cost-effective security framework allowing a distributed enforcement of access control policies using a set of policy enforcement points (e.g., SDN virtual switches). ACLFLOW takes as input a set of ACL rules, each one is composed of source/destination IP, source/destination port, a protocol, and a decision (i.e., allow of deny). Then, it uses a translation module that rewrites them into OpenFlow filter rules [MAB+08, SJK+14], which are then enforced by SDN virtual switches. To reduce the impact of the policy enforcement on the NFV service, ACLFLOW allows a dynamic prioritization of the most popular OpenFlow filter rules i.e., the rules affecting the highest traffic volume. Despite the fact that the contributions proposed in ACLFLOW mitigate the policy enforcement bottleneck of [YG15], ACLFLOW suffers form the following limitations. First, the authors did not provide any formal correctness proof to show that the ACL policy is correctly translated to a set of OpenFlow filter rules and that the deployed OpenFlow filter rules effectively enforced the set of ACL rules. Second, ACLFLOW is not suitable for enforcing more advanced and expressive access control policy models such as Role Based Access Control (RBAC), Organization-Based Access Control (ORBAC), Attribute-Based Access Control (ABAC), etc. Third, similarly to previous SDN-based approches, ACLFLOW cannot be used to enforce policies on an NFV as service architecture, as it needs to control and manage the NFV infrastructure to include the SDN framework.

Seeking to define a security management solution for NFV-based architecture, the authors of [Jae15] proposed an SDN based security orchestrator that (1) handles security on a hybrid Telco network i.e, configure the functions of the physical and virtual networks, (2) improves the ETSI NFV reference architecture with an extensive management of trust, and (3) offers a global view for fast and efficient topology validation. More specifically, the solution enables a security administrator to enforce specific hardening of the security of a network service. However, to completely meet end-to-end security, multiple responsibilities, such as security profiling, security policy management and automation, as well as trust management, should be considered in the proposed security orchestrator. Unfortunately, none of the previous functionalities has been defined in the paper. In addition, apart from the high level concrete use case and implementation of the provided security requirements are given. Moreover, this approach inherits the limitation of the previously discussed SND-based technique, namely, the fact that it can only be used to enforce policies on an NFVI as service architecture.

Matias et al. [MGT+15] presented an SND-enabled NFV approach allowing to achieve fine-grained control of which traffic is authorized to flow between the different entities (VNFs and other services) that compose a network virtual service. The proposed approach relies on three main components. First, an authentication service that allow authenticating the entities on a per service basis. Second, a classifier which is responsible for categorizing all the traffic exchanged between the involved entities according to the service to which the traffic is destinated. Third, a policy decision point responsible of determining whether the traffic should be allowed or denied. The previous decision is then enforced by the policy enforcement. Nevertheless, it is not clear if the proposed solution can handle policies expressed using advanced access control models such as RBAC, ORBAC, etc. In addition, it is difficult to assess the impact of the usage of this solution on the virtualized network service on which the access policy is deployed. Finally, as the solution is relying on SDN, similarly to the previous SDN-based solution, it can only be useful to enforce policies on an NFVI as service architecture.

The authors of [LQ16] proposed an SDN-based NFV orchestration framework called APPLE (Automatic aPProach for poLicy Enforcement). It enables two main functionalities. First, information flow policy enforcement meaning that the sequential order of VNFs that will be traversed by specific data flows should be respected. The previous requirement should be satisfied without the need to change the flow forwarding paths required by other network applications such as access control and routing. Second, resource isolation meaning that the resources used by any VNF should not be accessible to others, even if they are running on top of the same physical platform. In addition, APPLE includes an engine allowing to estimate the virtual network function demand for network-wide flows and proactively installs VNF instances for each potential flow to prevent the delays resulting from waiting for the new VNF instances getting started. While this approach can be used to enforce flow control policies efficiently, it is not designed to perform access control policy enforcement.

2.5.2 Orchestrator-based access Control

In this section, we discuss the security policy enforcement approaches on NFV infrastructure that define and/or rely on their own security orchestrators i.e., the approaches that are not based on the SDN framework.

In [PHMZ16], Montida et al. provide a comparative review on the existing NFV or-

chestration frameworks in an objective to analyze whether or not they can be used/extended to enable specifying, defining and enforcing security properties in the entire lifecycle of the NFV services. Then, the authors propose SecMANO, a security orchestrator that extends the MANO NFV orchestrator to allow the management of the security properties that needs to be ensured on virtualized network services. They extend the specification language of the Topology and Orchestration Specification for Cloud Applications (TOSCA) model [BBKL14] with particular high-level security attributes (e.g., access control attributes, privacy related attributes, confidentiality related attributes, etc.) that can be used by the security administrator to attach these security attributes to the virtualized network service components. They instantiate the proposed security orchestrator in [PHZ+18, PTH+17] through the implementation of an access control model allowing the specification and deployment of access control policies over NFV services. Nevertheless, the proposed approaches suffer from several limitations. First, it is not clear how the high-level security properties are refined/transformed to concrete-level security configurations. Without a clear description of such refinement/transformation, it is not possible to assess the correctness of the solution i.e., whether or not the concrete security configurations effectively deploy the high-level security requirements. Second, the TOSCA language extension allows only to handle simple RBAC access control rules and cannot be used to handle policies specified using more expressive access control models such as ORBAC. Finally, to deploy the access control policies, the solution requires policy enforcement points to be installed in each virtual machine hosting the NFV service components. The previous requirement makes the approaches proposed by Montida et al. [PHMZ16, PTH+17, PHZ+18] useful only when an NFVI as service architecture is considered.

The authors of [TCCM17] present a high-level architecture for NFV environments focusing on the automation of access control policy deployment. The proposed architecture involves three components: a management dashboard allowing to express the security requirements (i.e., access control policy) to be deployed; a policy engine allowing to translate the high level security requirement to security configuration that are deployed through a set of security mechanisms (i.e., packet filtering, intrusion detection, etc.); and a security monitoring module allowing to collect specific information (e.g., indicators of compromise) from associated security function enablers (e.g., intrusion detection). This collected information can be used by the policy engine to update the access control rules to mitigate specific new detected threats. Nevertheless, sim-

ilarly to [PHMZ16], no concrete high-level security requirement refinement method is proposed.

In [OKJ+17], Oh et al. introduce a generic architecture for Network Security Functions (NSF) based security management on NFV infrastructures. The main goal is to enable any NFV-based security application managed by a security controller to be seamlessly integrated into NFV-enabled networks. The proposed architecture integrates new components to the ETSI NFV architecture enabling security policy management. First, the NSF client components allow generating high-level security policies that need to be enforced to mitigate security attacks. Second, a security management component responsible for refining the high-level policies to concrete-level configurations/rules that can be enforced by available NSFs, such as firewalls and intrusion detection systems. Thanks to this architecture, application users can enforce their high-level security policies in a user-friendly manner. However, despite that the authors briefly discussed the usage of the proposed framework to fulfill specific high-level security requirements, such as dangerous domain blacklisting, time-based access control policies enforcement, and suspicious call detection for VoIP-VoLTE services, no formal high-level policy refinement method is proposed, which make evaluating the correctness of the generated concrete level configuration not possible.

To mitigate the security threats raised by the lack of access control in NFV services, Guija and Siddiqui [GS18] propose an architecture integrating well-known solutions allowing (1) the authentication of the entities/components that interact with the NFV service and (2) th authorization of the different actions/events that are performed in virtualized services. Concretely, the proposed approach relies on the NFV-based SONATA Service Platform [DKP+17] for implementing authentication and authorisation mechanisms, specifically for identity and access control of micro-services in 5G platforms for services virtualisation, orchestration and management. This solution uses OAuth 2 [Har12] and OpenID Connect [SBJ+] to form the implementation of the user management module allowing to enforce RBAC policies as well as identity management in a centralized authorization approach. Unfortunately, this dependency on OAuth 2 and OpenID Connect makes this solution usable only with virtualized network services where an HTTP-based communication is used between their different components.

In [BVL+19], Basile et al. proposed an approach for automatically enforcing security policies in NFV networks. It uses a high-level policy language [BLP+15] allowing to express the security requirements that need to be deployed in an NFV architecture.

The used high-level policy language is an authorization language in which policy rules are represented with natural language sentences. An example of rule is "SSH traffic should not be allowed between data processing and data storage subnets". Then to be deployable, the high-level rules need to be refined into concrete rules and configurations. The proposed approach defines a policy refinement module called Policy Manager. The latter is used to perform a transformation of user-specified, high-level, and natural language security policies into low-level configuration of virtual network functions. The refinement is performed in two steps. First, using a set of inference rules, the high level sentences that compose the high-level policy rules are mapped to low-level concepts. For example, a "data processing subnet" sentence is mapped to a range of IP addresses associated with the VNF instances that are processing (e.g., encoding, encrypting, etc.) the data. In addition, the sentence "should not be allowed" is mapped to a filtering prohibition. Second, once the high-level policy is refined to a concrete one, the proposed approach chooses then for each concrete policy rule, the best network security function that can be used to enforce the rule on NFV services. Nevertheless, this solution suffers from two main weaknesses. First, the correctness of the used refinement technique relies strongly on the correctness of the inference rules that will be used to map high-level requirements to deployable rules. Unfortunately, neither the set of inference rules to be used nor their correctness are provided in [BVL+19]. Second, the authors do not consider conflict management in their solution. Conflicts can occur at low-level (deployable) policies as well as between the network security functions that will be used to enforce the policies.

Murillo and Rueda [MR20] presented an access control policy enforcement framework for NFV-based industrial control system (ICS). The framework defines a new security policy specification language that allow security administrators to easily model, in one hand, virtual ICS systems with various components such as sensors, actuators and PLCs, as well as the different types of interconnections that are used in the ICS systems; and in the other hand, the security requirements that should be enforced in the system. Then, the proposed framework relies on a policy compiler module to translate the high-level security requirements to concrete ones depending on the specification of the target ICS. Finally, the framework relies on a policy enforcement module that is in charge of intercepting the requests generated by the different entities in the ICS system (e.g., users, servers, sensors, etc.) and enforce the concrete security policy by accepting/denying those requests. Same as the previously discussed solutions,

the current does not formally specify how high-level security requirements are translated to concrete ones. In addition, it is difficult to evaluate the efficiency of the policy enforcement strategy as it is not clear where the policy enforcement points are to be installed in the target ICS system.

To sum up, all the previously discussed security policy enforcement solutions suffer from at least one of the following limitations. First, lack of generalisability meaning that the proposed approaches [PHMZ16, PHMZ16, PHZ+18, GS18, MR20] can only be used to enforce specific access control requirements e.g., ACL rules and RBAC rules. Second, the lack of formalization which makes the evaluation of the correctness of the proposed solutions[1] not possible. Third, the lack of conflict management[1] that may occur between either two or many security requirements or between security requirements and the functional requirements that the virtualized service has to fulfill (see Chapter 4 Section 4.3.2). If these conflicts are not resolved, then the correctness of the policy enforcement cannot be guaranteed. Fourth, the non compliance with the standard ETSI NFV architecture (e.g., the approaches proposed in [PHMZ16, PTH+17, PHZ+18] as additional components need to be added to the NFV infrastructure to allow security policy enforcement. Finally, none of the previously discussed approaches has considered the optimization of the access control policy enforcement cost in terms of the resources required by the enforcement points and the impact on the functionalities provided by the virtualized network service.

2.5.3 Optimal Deployment

NFV services aim to bring agility, flexibility, and high performance for virtualized infrastructures. The enforcement of access control policies on NFV services relies on one or multiple enforcement points that, on one side, require computation and storage resources to deploy the concrete rules/configurations on the NFV service, and on the other side may impact the performance of the latter. Considering the above problem, several approaches in the literature have investigated the optimization of the required resources and/or the impact on performance when access control policies are deployed in NFV services. These approaches can be classified into two classes:

- First, solutions that investigate the impact of the number and the placement of the policy enforcement points in the NFV infrastructure on the amount of required

1. all previously discussed approaches suffers from this limitation

resources and the performance of the provided network services.

- Second, rule/configuration placement based approaches that investigate the impact of the placement of access control policies' rules in the NFV infrastructure on the amount of required resources and the performance of the provided network services.

Policy Enforcement Point Placement

Lee et al. [LPS13] investigates the problem of finding the best policy enforcement point placements while taking into consideration the bandwidth constraint on each link. To enforce the access control policies, the authors propose to filter the communication between different entities that compose the virtualized infrastructure. To accomplish this filtering, the data flows are reroute to pass through one or several firewall instances. Then, the authors proposed a greedy algorithm to find the best placement of the firewall instances to reduce to the best the traffic congestion on the virtualized network service. Nevertheless, the proposed approach has several limitations. First, it does not consider optimizing the resources that are to be used by the firewall instances. Second, it is not clear how the number of firewall instances is decided in the system. Third, the proposed approach can support only the deployment of ACL policies.

In [BMSV20], Bringhenti et al. proposed an approach defining a methodology allowing to automatically allocate and configure channel protection systems in virtualized networks. Channels are virtual links that ensure specific security properties (e.g., access control, confidentiality, integrity) to protect highly sensitive information transiting through the NFV infrastructure. The proposed approach relies on the formulation of the studied problem as a MaxSMT problem. The authors generalized their approach in [BMSV21] to study the optimal placement of security functions (e.g., access control) called virtual Network Security Functions (vNSFs) in a virtualized network.

Bringhenti et al. [BMS+20] addressed the problem of automatically computing the optimal allocation scheme and configuration of virtual firewalls on a virtualized infrastructure. The proposed approach relies on the dynamicity of the definition and configuration of network services offered by emerging technologies such as NFV and SDN technologies. This dynamicity of the network configuration is used to replace manual and error-prone tasks such as, network isolation in case of cyberattacks and firewall rule enforcement by automatic approaches, which relies on placing, configuring

and enforcing NFV. If previous tasks are manually realized, it can cause non-correct and/or non-optimal policy enforcement. For that, it is necessary to replace manual and error-prone tasks by automatic approach. To meet the previous objective, the authors proposed a new methodology for automating and optimizing the assignment and the configuration of virtual firewalls on virtualized networks. They used as optimization criteria the minimization of the number of assigned virtual firewall instances as well as the minimization of the number of rules inside each firewall configuration and they used as evaluation criteria the number of firewall allocation places and the number of ensured/violated network security requirements. The strategy adopted to reach both formal correctness and optimality consists of formally modelling the problem as partial weighted Maximum Satisfiability Modulo Theories (MaxSMT) problem. The main drawback of the approach proposed by Bringhenti et al. [BMS+20] is that it does not take into consideration the optimization of the impact (e.g., latency, traffic overhead, etc.) of the deployment of the access control policy on the virtualized network, which may result in quite inefficient enforcement (e.g., high latency and/or low bandwidth) of the access control policy.

Rules placement

In [LQ16], Li and Qian propose an orchestration framework based on SDN/NFV called APPLE (Automatic aPProach for poLicy Enforcement). It installs instances of virtual network functions (VNFs) automatically to enforce the network function policies. The proposed framework offers three preferred properties such as Policy enforcement, interference freedom, and resource isolation. APPLE is able to make an estimation of the NF demand for network-wide flows and proactively installs VNF instances for each potential flow to prevent delays in getting started. APPLE uses an optimization engine to identify the placement of VNFs and a flow marking scheme to reduce TCAM (Ternary Content-Addressable Memory) consumption. However, APPLE does not provide any formal proof of the correct enforcement of the access control policy.

In [MRD18], Mauricio et al. focus on the problem of filtering traffic on cloud computing. In virtualized environment, ACLs are traditionally used as a straightforward way to enforce traffic filtering policies. Nevertheless, depending on the size of the considered system, the number of ACL rules to be enforced may be large, which may not be very adapted with the storage and processing capacities of the policy enforcement points (e.g., TCAM virtual router). To solve this problem, the authors proposed a cost-effective

NFV/SDN security framework named ACLFLOW. The latter allows a transformation of regular security rules (source/destination IP, source/destination port, and protocol) into OpenFlow filtering rules. It builds and monitors distributed OpenFlow using routers for controlling virtual machine traffic in a cloud computing environment. It optimizes switching operations by using an algorithm that dynamically prioritizes the most popular rules. The solution relies on the fact that based on the set of ACL rules to deploy, the exchanged traffic is monitored to know which rules have the highest priority and translate them into a control plan. As evaluation criteria, the authors considered the maximum HTTP request rate, the maximum throughput, and round-trip time. Nevertheless, no formal modeling was proposed in the paper, which makes checking the correctness of the deployment of the access control policy not possible.

The problem considered by Amin et al. in [ASM19] is to find the optimal method to place Access Control Lists (ACL) that eliminates the path misconfigurations[2] and unwanted traffics. To secure a traditional computer system, network operators typically implement fine-grained network policies that are known as ACLs at switch and/or router network interfaces through the use of low-level commands. Network performance can be degraded if there is a misconfiguration that allows unauthorized users to access confidential resources. In general, the implementation of ACL policies is performed manually which make it extremely difficult and costly when dealing with large infrastructure. For that, the authors proposed a systematic design approach to optimize the implementation of access control policies in hybrid SDN networks. The solution relies on decision tree and K-partite graphs [LWZY06] to search for the best placement of ACL policies. The authors use as optimization criteria, the minimisation of the number of ACL rules and the number of transmission of unwanted traffic. As evaluation criteria, the authors consider the computation time of the K-partite graphs, the traffic optimization, the number of ACL rules versus number of firewalls, end-to-end delay calculation, and the packet delivery success rate. Similarly to the previous approaches, no formal modeling was proposed in the paper, which makes checking the correctness of the enforcement of the access control policy rules not possible.

Kim et al. [KYNL20] considered the problem of optimizing and deploying firewall rules in Software-Defined Networks. Since the traffic inspection capacity of a firewall is restricted by the network processor speed, memory capacity, and power consump-

2. a misconfiguration can drastically degrade the network performance by enabling unauthorized users to access confidential resources

tion, large-scale network traffic must be inspected by multiple firewall instances. The placement of this firewall in the infrastructure may have an impact on the optimal deployment of access control policies. So, it is necessary to study the impact of firewalls placement on the optimization of the access control policies deployment. For that, the authors proposed a software-defined firewall system that relies on SDN functionalities to dynamically enforce access control requirements through firewall operations. The authors propose to optimize the overall volume of data traffic in the network by finding the appropriate OpenFlow-enabled switches in which the SDN flow rules should be deployed, while respecting the processing and storage capacities of OpenFlow switches. The authors used the greedy algorithm to solve the problem. They used as evaluation criteria the service throughput as a function of number of rules and the resource as a function of number of rules. However, the authors do not consider the optimization of the impact e.g., in terms of the latency introduced by the access policy enforcement on the virtualized service.

Bringhenti et al. [BMSV21] proposed a new approach to automatically configure and deploy access control requirements in a virtual service graph. In contrast with the previous approaches, the authors consider that several policy enforcement mechanisms can be used. The goal is then, not only to optimally enforce the access control requirements but also to choose the right concrete mechanisms that can be used to enforce them. To meet the previous objective, Bringhenti et al. introduce a new level of abstraction in which a set of functionalities (i.e., function features) that can be used to enforce each security requirement (e.g., access control rule). Then based on the identified functionalities and their possible configurations, the approach chooses the optimal combination of concrete policy enforcement mechanisms and their placement in the virtual service graph. The proposed optimisation algorithm is multi-objective that optimizes physical resources such as CPU, Memory, and Bandwidth. Unfortunately, they do not consider the optimization of the impact (e.g., introduced latency) of the chosen policy enforcement strategy on the virtual service.

In [JJW+21], Jiang et al. have considered the problem of access control rule placement in multi-tenant virtualized infrastructure. Such infrastructure needs usually many fine grained policies to be deployed to enforce specific security requirements. The authors point out that in case in which the access control policies have to be deployed in large scale virtualized infrastructure, millions of rules will need to be deployed. To solve this problem, they provide an online rule placement (ORP) algorithm to minimize

traffic overload based on a virtual Cloud Rule Information Base (vCRIB). The solution relies on the fact that the online algorithm is able to check the traffic matrix update after the adjustment of each rule subset. The authors used as optimization criteria the minimization of the number of used rules, the maximization of the amount of carried traffic, and the minimization of energy consumption and as evaluation criteria the optimization algorithm time. Similarly to the previous approaches, the authors do not consider the optimization of the impact (e.g., introduced latency) of the access policy enforcement on the virtual service.

The Table 2.2 gives a comparative view on the approaches that consider the optimization of the deployment of access control policies on virtualized environments. For each proposed approach, we identify the optimization criteria, the optimization strategy (policy or rule level optimization), and the optimization algorithm that has been used. According to the Table 2.2, five out of the eight studied approaches have investigated the optimization of the resources used for access control policy deployment, and only three (resp. one) have (resp. has) considered the optimization of the impact in terms of traffic overhead (resp. latency) on the target virtualized service. Moreover, none of the studied solutions has considered both the optimization of the required resources and the introduced impact due to the enforcement of the access control policy.

2.5.4 A Comparative Overview

In this section, we provide a comparative view of the different approaches discussed above. The comparison is performed in the basis of the following properties.

Correctness: To allow an easy specification of the security requirements to be enforced, most security policy enforcement approaches discuss above proposed high-level security requirements specification languages/methods. These high-level requirements are then translated to concrete security configurations/rules that are to be enforced on the considered system. The correctness property requires the concrete security configurations/rules to be enforced to effectively and correctly enforce the specified high-level security requirements. In other words, once the concrete security configurations/rules are enforced on the target system, no high-level security requirement can be violated by malicious adversaries.

Paper	Optimization Criteria					Optimization Strategy		Optimization Algorithm
	TO	R			L	RP	FP	
		M	CPU	B				
[BMS+20]	✗	✓	✓	✓	✗	✗	✓	MaxSMT
[BMSV20]	✗	✓	✓	✓	✗	✗	✓	MaxSMT
[LPS13]	✓	✗	✗	✗	✗	✗	✓	Approximation algorithm and Greedy algorithm
[MRD18]	✗	✓	✓	✓	✗	✗	✓	ACLFLOW prioritization algorithm
[ASM19]	✓	✗	✗	✗	✓	✓	✗	Decision Tree Construction Tree Traversing
[JJW+21]	✓	✗	✗	✗	✗	✓	✗	Online rule replacement (ORP) algorithm
[BMSV21]	✗	✓	✓	✓	✗	✗	✓	MaxSMT
[KYNL20]	✗	✗	✓	✓	✗	✓	✗	Greedy algorithm

Table 2.2 – Comparative view of optimization approaches. We used TO to denote traffic overhead, R to denote resources, L to denote latency, M to denote memory, and B to denote bandwidth

Adversary Model. In existing access control policies enforcement approaches, two types of adversary models can be considered. First, external adversaries who represent malicious entities which are trying to violate the access control policy by interacting remotely with the target infrastructure. Second, insider adversaries which are malicious entities that are controlling one or several components of the target infrastructure and aim to use them to bypass the deployed access control policy.

Efficiency: The enforcement of access control policies usually requires the analysis of the traffic flowing inside the system on which the policy is to be deployed to decide whether the traffic should be allowed or denied. This will inevitably introduce a latency at the network level of the target system. The efficiency property describes how the aforementioned latency increases as the number of rules/requirements in the policy to be deployed and the complexity of the target system (e.g., in the case of NFV service: the number of VNF, the number of forwarding paths, etc.) increases. Concretely, we say that an access control enforcement approach is efficient if the introduced latency grows (sub) linearly with the size of the policy and the complexity of the target system, and inefficient otherwise.

Optimality: Access control policy enforcement does not only introduce a network-level latency, but it also requires computation and storage resources that will be used by policy enforcement points to analyze and authorize the traffic flowing through the target system. The previous two elements can be impacted by several enforcement decisions such as, the location of the enforcement points inside the target system, the rules that will be enforced by each enforcement point, etc. The optimality property is used to determine if an approach offer an (near[3]) optimal deployment of access control policies allowing to find the best possible trade-offs between the impact in terms of latency resulting from the deployment of the access control policy and the used resources.

High Expressiveness: Several access control policy specification models have been proposed in the literature such as, RBAC, ORBAC, ABAC, etc. Each of them brings the ability to model different and/or more fine-grained access control requirements. The high Expressiveness property is used to determine if an existing access control deployment approach offers sufficiently expressive specification language to support the

3. Some of the existing approaches do not explore all the search space to find the optimal solution.

enforcement of most well known policies specification model such as RBAC, ORBAC, ABAC, etc.

Conflict Management: Most of the exisiting access control models allow defining both authorization (positive) and prohibition (negative) rules. Using the previous, one can specify one of the following three policies:

- Open policy: the policy is composed only of prohibition rules. Only the accesses that are explicitly prohibited by these rules should be denied.
- Closed policy: the policy is composed only of authorization rules. Only the accesses that are explicitly stated by these rules should be allowed.
- Mixed policy: the policy is composed of both authorization and prohibition rules.

It has been shown in the literature by Alfaro et al. [ACCB07] that open and closed policies lead to complex concrete-level configuration/rules when excluding some specific cases of general rules that should always apply, which can be avoided by using mixed policies. However, the latter may introduce conflicts that occur between prohibition and authorization rules. These conflictual rules should be appropriately addressed to ensure a correct deployment of access control policies. The conflict management property is used to denote whether an access control policy enforcement approach can correctly deploy mixed access control policies containing exceptions and/or conflict rules.

Easy deployment: As we have seen previously in this section, some approaches require the modification of the infrastructure in which the target system is running e.g., the modification of the ETSI NFVI architecture by adding new components. Therefore, the easy deployment property is used to denote whether an approach is easily deployable on an NFV service or not. Concretely, we say that an approach allows easy deployment of access control policy if it does not require the modification of ETSI NFVI infrastructure. For example, approaches based on SDN often require the modification of the ETSI NFVI to add the components (e.g., SDN controller) that enable specific SDN features. Hence, these approaches are considered to provide hard deployment of access control policies on NFV services.

Paper	Properties						
	Adversary Model	**Efficiency**	**Correctness**	**Optimality**	**High Expressiveness**	**Conflict Management**	**Easy deployment**
[LPS13]	Outsider	-	✗	✓	N/A	N/A	✓
[BLP+15]	Outsider	-	✗	✗	✓	✓	✓
[YG15]	Outsider	● ● ●	✗	✗	✓	✗	✓
[Jae15]	Outsider	-	✗	✗	✗	✗	✓
[MGT+15]	Outsider	-	✗	✗	✗	✗	✗
[PHMZ16]	Outsider	-	✗	✗	✗	✗	✓
[LQ16]	Outsider	● ○ ○	✗	✗	✗	✗	✗
[TCCM17]	Outsider	-	✗	✗	✓	✗	✓
[PTH+17]	Outsider	-	✗	✗	✓	✗	✓
[OKJ+17]	Outsider	-	✗	✗	✓	✗	✓
[GS18]	Outsider	● ● ○	✗	✗	✗	✗	✓
[PHZ+18]	Outsider	● ● ○	✗	✗	✓	✗	✗
[MRD18]	Outsider	● ● ●	✗	✓	✗	✗	✗
[ASM19]	Outsider	● ● ●	✗	✗	✗	✗	✗
[BVL+19]	Outsider	-	✗	✗	✓	✗	✓
[BMSV20]	Outsider	-	✗	✓	✗	✗	✓
[KYNL20]	Outsider	● ● ●	✗	✓	✗	✓	✗
[MR20]	Outsider	-	✗	✗	✓	✗	✓
[BMSV21]	Outsider	● ● ○	✗	✓	✗	✗	✗
[JJW+21]	Outsider	● ● ●	✗	✓	✗	✗	✓

Table 2.3 – Comparative view of optimization approaches.

Table 2.3 gives a comparative review of existing access control deployment solutions regarding the properties defined above. We use N/A to denote that the property is not applicable to the solution. In addition, we use - to denote that the information given in the solution does not allow to state if the latter satisfies a given property. Finally we used '• • •', '• • ○', and ′ • ○○′ to denote respectively linear, super-linear, and exponential complexity.

According to Table 2.3, we can make the following observations.

1. All existing approaches consider only outsider adversaries who are aiming to bypass the enforced access control policy by interacting remotely with the target infrastructure. None of them has investigated the enforcement of access control policies in the presence of an insider adversary who compromises part of the virtualized infrastructure.

2. None of the proposed solutions has provided formal evidence proving the correctness of the used access control policy refinement and enforcement methods.

3. Almost half of the proposed solutions do not provide any theoretical and/or experimental results making evaluating their efficiency not possible. Particularly, only a quarter (5 out of 20) of the studied solutions have reported results showing their efficiency.

4. Few of the studied approaches have proposed methods for finding the optimal access control policy enforcement solution. Moreover, as shown in Table 2.2, all previous approaches have consider either the optimization of the resources (e.g., CPUs, memory) used to enforce the policy, or the impact (e.g., the latency introduced by the policy enforcement points) on the virtualized infrastructure due to the enforcement of the policy. None of the proposed solutions has investigated both the optimization of the required resources and the introduced impacts.

5. Around half of the studied approaches enable high expressiveness of the access control policy to be enforced while proposing a lightweight deployment strategy (i.e., without requiring the modification of the virtualization infrastructure on which the virtualized network service is running). However, only two of the proposed solutions have investigated the management of conflictual rules in the policy to be enforced.

6. Finally, we can clearly observe that none of the studied approaches has been able to fulfill all the properties that we have identified.

2.6 Open Challenges

According to the aforementioned observations, the following challenges need to be addressed.

- The definition of a formal model for access control policy enforcement in NFV services. The model should be sufficiently expressive to allow to express high level access control requirements modeled using well known access control models. In addition, it should allow a provably correct refinement of the high level access control requirements towards concrete-level rules that are compliant with ETSI-NFV architecture.

- The definition of an access control enforcement strategy that allow to optimize both the introduced latency and the required computation and storage resources that will be used by policy enforcement points to enforce the access control requirements.

- The definition of an access control enforcement strategy that can deal with insider adversaries that are controlling one or several unknown components of the target virtualized service. Concretely, the proposed model should prevent insider adversaries from violating the enforced access control requirements.

- High-level access control policies are often composed of a set of authorizations and prohibitions, which may give place to conflicts and exceptions in the policy to be enforced. Consequently, the definition of a solution allowing efficient enforcement of high-level mixed access control policies containing exceptions in virtualized infrastructure is required.

- The design and development of a simulation framework allowing to evaluate the effectiveness and efficiency of the deployment of access control policies on NFV service.

2.7 Conclusion

In this Chapter, to answer the raised research question, we conduct a literature review of existing and potential security threats in NFV architecture and existing solutions for enforcing access control policies on virtualized infrastructure.

We answer the research questions RQ-1 and RQ-2 by conducting an extensive review of the security threat in NFV-based infrastructure. Then, we classify them according to the concerned NFV layer.

Answering RQ-3 and RQ-4 were not evident due to the fact that an in-depth study of the access control enforcement in virtualized environments was missing in the literature. To answer RQ-3 and RQ-4, we conduct an extensive literature review on access control policies on virtualized infrastructure. In synthesis, we found 20 distinct solutions. We compare them according to the set of properties that need to be considered when designing an access control policy enforcement on virtualized infrastructures. These properties include the high expressiveness of the model, the correctness of the refinement and deployment strategies, the optimality on both the required resources and the introduced latency, the ability to deal with insider adversaries, and the effective management of conflicts in the policy to be deployed.

The conducted comparison allows us to make the following observations. First, only nine of the existing solutions present evidence (e.g., simulations and experimentation) about their real-world usage. Second, none of the existing solutions provides formal proofs on the correctness of their refinement and enforcement methods. Third, very few of them provide high expressiveness for security requirements, enable conflict management, and/or ensure an easy deployment of access control policies on virtualized infrastructure. The previously mentioned shortcomings of the existing solutions raise several open challenges that are to be addressed by investigating the following research question.

> **RQ-5:** How to create **efficient**, **high expressive**, **correct**, and **easy deployable** approach allowing to **optimally deploy mixed** and **complex** access control policies on virtualized infrastructures?

The previous question guides, for the most part, the rest of this dissertation. We will break RQ-5 down into several research questions related to the properties that we aim to ensure, and provide some solutions that answer the identified research questions.

CHAPTER 3

A DOMAIN TYPE ENFORCEMENT OF ACCESS CONTROL POLICIES IN NFV SERVICES

Contents

3.1 Introduction

As we have seen in Chapter 2, several access control policies deployment approaches in virtualized infrastructure have been proposed. Based on Table 2.3, we can see that most of these approaches are suffering from at least one of the following limitations. First, none of the existing approaches provide formal proofs supporting the correctness of the deployment of access control policies. Second, most of the existing approaches rely on specification models that are not expressive enough to handle high level policies specified using other access control models such as RBAC, ABAC, and ORBAC. Finally, several existing approaches require the modification and the management of the NFVI which make them non-compliant with the ETSI-NFV architecture and thus not easily used in practice. Hence, to overcome the previous limitation, in this chapter, we address the following research questions:

- **RQ-5.1:** How to define a *sufficiently expressive high-level* specification model that allows to *correctly* express policies specified using other well known access control models such as RBAC, ABAC, and ORBAC?
- **RQ-5.2:** How to correctly refine the high level access control policy towards concrete-level deployable requirements?
- **RQ-5.3:** How to define an ETSI-NFV compliant deployment of the concrete-level requirements?

To answer the previous research questions, in this chapter, we propose a formal model that provides a software-defined access control as a service capability for virtualized network services. First, we define an expressive specification model to be able to express high-level access control requirements to be enforced over virtualized network services. Then, we show that our specification model can correctly express access control requirements specified using other access control models such as RBAC, ABAC, and ORBAC. Second, we propose a provably correct method for refining high-level access control requirements towards a domain type enforcement (DTE) concrete specification. Finally, our model defines an ETSI-NFV compliant and efficient enforcement method, as illustrated by the different conducted experimental evaluations in Section 3.4. In addition, our policy enforcement method is scalable as it is possible to add as many enforcement points as needed (e.g., for load balancing purposes) without impacting the functioning of the network services.

This chapter is organized as follows. Section 3.2 provides some background on NFV service, its composition and how they are specified according to the ETSI NFV standards. Section 3.3 defines our access control policy enforcement model. Section 3.4 provides an overview of the implementation of our model and presents the evaluation results. Finally, Section 3.5 concludes this chapter.

3.2 Background

This section provides background material about all main technologies to enable the deployment of our security policy.

3.2.1 Virtual Network Service

A Virtual Network service is an interconnection of virtual network functions (VNFs) to provide a desired functionality. VNFs refer to the abstraction in software of network resources traditionally provided in hardware. It can combine multiple physical networks into a virtual software network, or divide a physical network into multiple independent and distinct virtual networks. Common VNFs include virtualized firewalls, routers, and network address translation (NAT) services. Most VNFs are run in virtual machines (VMs) on common virtualization infrastructure software such as VMWare or KVM. It is necessary to chain the traffic through the VNFs in an ordered graph to apply the network services. In the case of NFV, the chaining is called the Virtualized Network Function Forwarding Graph (VNFFG) to take into account the virtualization on an overlay network. A NFV service is composed of the following elements:

- A set of VNFs: A VNF represents a Network Function (NF) software implementation which can be deployed on an NFV infrastructure [E$^+$14].
- A set of VNF Forwarding Graph: It represents a logical graph that defines the connectivity between the network function such as VNF and the connection point in order to define a traffic flow between them. This notion [VNF] is defined by ETSI. It is known also as Service Function Chaining (SFC) that provides a level of abstraction so that the operator can compose the network services in real time. It is used to manage traffic through a sequence of network functions (NF) that should be traversed in an order list of VNFs according to different routing

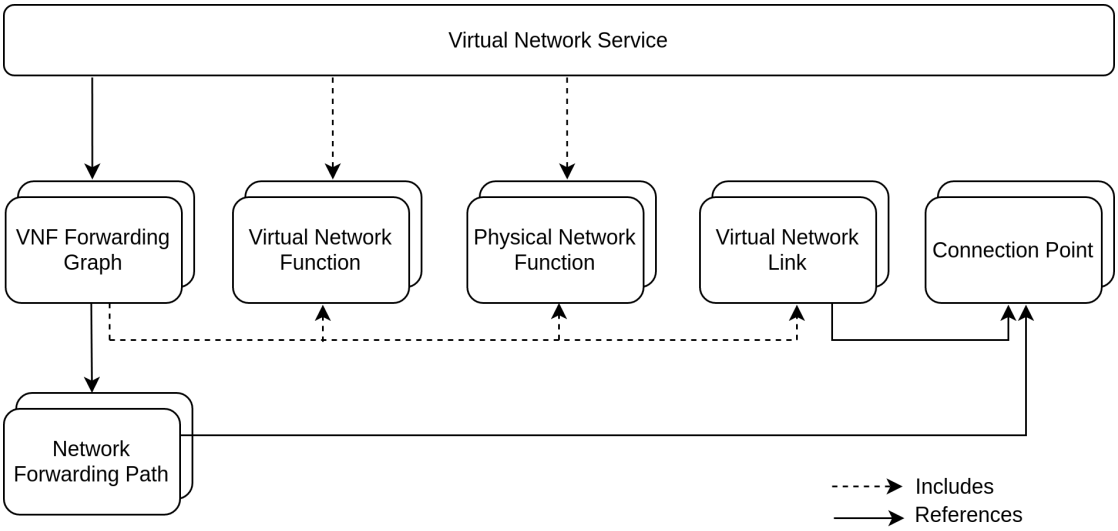


Figure 3.1 – Network Service Descriptor overview

policies. This latter is a sequence of connection points through which the packets traverse. Each forwarding graph is composed of a set of forwarding paths.

- Physical Network Function: It is an implementation of a Network Function by a closely coupled software and hardware system.
- Virtual Links: A virtual link (VL) is a logical connection used to interconnect the VNFs. VL defines the connectivity between connection points and any associated target performance metrics (e.g. bandwidth, latency) [E$^+$14].

To specify the configuration of the network service, ETSI uses network service descriptors that describe the requirements and attributes for deployment and lifecycle management of an entity that are defined below:

- VNF descriptors (VNFD) are templates that define the instances to be implemented and the resource requirements of each instance (CPU, memory, interface and network). The descriptor also defines the types of interfaces with the NFVI infrastructure and the expected KPIs.
- VNFFG descriptors (VNFFGD): VNFFG are described by VNF Forwarding Graph Descriptors (VNFFGD). It is a model that specifies the requirements for the connectivity of connection points that are attached to a physical network function.The VNFFGD contains metadata about the routing graph of the VNF virtualized network functions, as well as references to the infrastructure containers, the VNF

instances and the link between the VNFs. The metadata also contains policy rules for the routing functions (routing and switching rules).

- Physical Network Function Descriptor (PNFD): is a model that defines the requirements for the connectivity of the connection points attached to a physical network function.

- Virtual Link Descriptor: It is a specification model that describes a Virtual Link within an NFV (Network Functions Virtualization) system.

3.2.2 Domain and type enforcement (DTE)

Domain and Type Enforcement (DTE) [BSS+96] was developed by Badger, et al. in 1996. The objective was to enhance the ease and suitability of the flexible and robust Type Enforcement (TE) access control mechanism. It protects the integrity of military computer systems and was designed to be used in combination with other access control techniques. As with many access control schemes, DTE views a system as a collection of active entities (that represent the subjects that are going to perform actions) accessing a collection of passive entities (that represent the objects over which the action are performed) based on rules defined in attached security context. Each subject of the system belongs to DTE domain and each object of the system is associated to DTE type. The access policy is then defined as:

- A set of actions that can be performed by a set of entities belonging to a domain on the set of objects associated to a type.

- A set of domains transitions defining the conditions under which a subject can transit from a domain to an other.

Based on Type Enforcement, a global table called Domain Definition Table (DDT) contains allowed interactions between domains and types. Each row of the DDT represents a domain, and each column represents a type. In addition, It manages

- Access Control by controlling access from domains to types

- Flow Control by controlling transitions from domains to other domains

In DTE abstraction, a domain definition specifies the rights that each domain has over objects of a given type. Any subject in the same domain has the same set of access rights. Assigning subjects to domains gives a flexible mechanism for implementing a security policy. All subjects associated with ordinary users can be represented by a domain. This latter does not have to be associated with the rights of a single user. Many other security models grant rights according to certain characteristics of a user, such as his or her security clearance (multi-level security), role (i.e., function) in the organization (role-based access control) or identity (standard UNIX access control). One of the main advantages of DTE is that it lets the administrator strike an adequate compromise between security and policy complexity. DTEL is a simple high-level symbolic language used by DTE to express reusable DTE configurations. It provides four primary statements for expressing DTE configuration for a single host:

- The type statement declares one or several types as part of the DTEL configuration. For each object in the target system, only one of the types is associated.
- The domain statement which is composed mainly of the following three elements:
  - Subject access rights: are permissible access modes (e.g., use to perform specific action) to subject in other specified domain.
  - Object access rights: are permissible access modes (e.g., write, read, delete) to object of specified types.
  - Entry points: In the context of an NFV system, we are not handling processes and files, we are rather dealing with virtual services. In an abstract way, an entrypoint is a concept that will allow the transfer of an entity from one DTE domain to another.
- The initial-domain statement: It associates to each subject that joins the system an initial domain to which the subject will belong. For example, a subject representing a HTTP client will be placed in the HTTP domain.
- The assign statement: is utilized to assign exactly one type with each object on the target system. Based on directive hierarchies, it supports a technique "implicit typing", to associate types with objects.

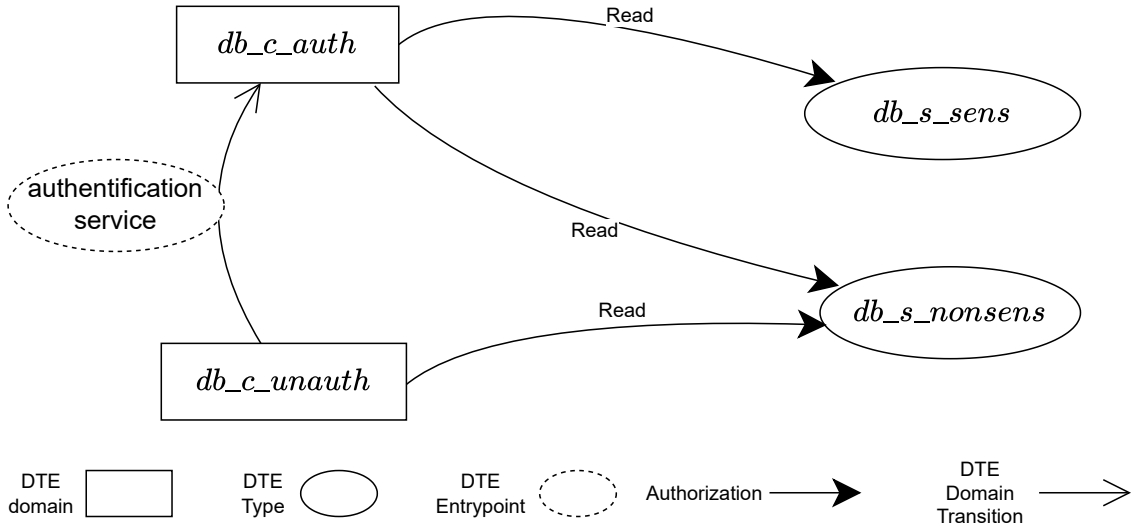


Figure 3.2 – DTE specification of the access control policy used in Example 3.1.

Example 3.1 *To illustrate access policy enforcement using DTE, let us consider a system composed of a database client* db_c *and two database servers* db_s_1 *and* db_s_2*. Let us consider an access control policy composed of the following two rules:*

- ***Rule 1:*** db_s_1 *contains non-sensitive information that can be accessible by both authenticated and non-authenticated database clients.*
- ***Rule 2:*** db_s_2 *contains sensitive information that should be accessible only to authenticated clients.*

The previous policy can be specified in DTE as depicted in Figure 3.2 and described in the following:

- *Using type statements, we create two DTE types* db_s_sens *and* $db\_s\_nonsens$*. Then, using assign statements, we assign* db_s_1 *to* $db\_s\_nonsens$ *and* db_s_2 *to* db_s_sens*.*
- *Using domain statements, we create two domains* db_c_unauth *and* db_c_auth*. Using object access rights, we specify that subjects in* db_c_unauth *can read the objects in* $db\_s\_nonsens$*, while subjects in* db_c_auth *can read objects in both* db_s_sens *and* $db\_s\_nonsens$*.*
- *We create a transition domain (a DTE entry point) represented by an authentication service allowing database clients to transit from* db_c_unauth *to* db_c_auth *as soon as they are authenticated.*

- *Finally, we use an initial domain statement to specify that any new instance of a database client should belong to db_c_unauth (the domain containing the unauthenticated database client).*

According to the previous DTE specification, a non-authentication database client will be initially placed in db_c_unauth. The latter can only access objects in $db\_s\_nonsens$ (databases containing non sensitive information) which satisfies the rule 1. In addition, once authenticated, a database client can transit to the domain db_c_auth which allows access to objects associated to both $db\_s\_nonsens$ and db_s_sens types, which satisfies the rule 2.

3.2.3 Access control models

Several access control models have been defined in the literature. In the following, we introduce the most known models with their access control requirement specification capabilities. The content of this section will be helpful later in this chapter to show that the requirements specified using any of the following models can be correctly translated to our access control policy specification model (see Section 3.3).

Role-Based Access Control (RBAC)

RBAC is a model of access control where permissions are associated with roles, and users are assigned to appropriate roles [SCFY96] (Figure 3.3). In this model, users acquire permissions by being members of roles [FSG+01]. Defined user roles represent the work processes in an organization and vary from company to company. The breakdown of user roles can be done by department, location, cost center, or (more common) employee responsibilities.

Pioneered in the 1970s, this model has gained acceptance and it is the de facto standard access control model for many systems having originated many derivatives. It has been accepted as the best-practice model, and can offer many advantages: flexibility, reduced administration work (as individually assigning permissions is not an issue), less error prone, increased efficiency and transparency. Yet, the setup of such a model is labor-intensive (especially in what concerns the translation of organizational structures). It is not the best model for temporary assignments (as there is not the concept of time lease at the concept). Additionally, in small companies, the process

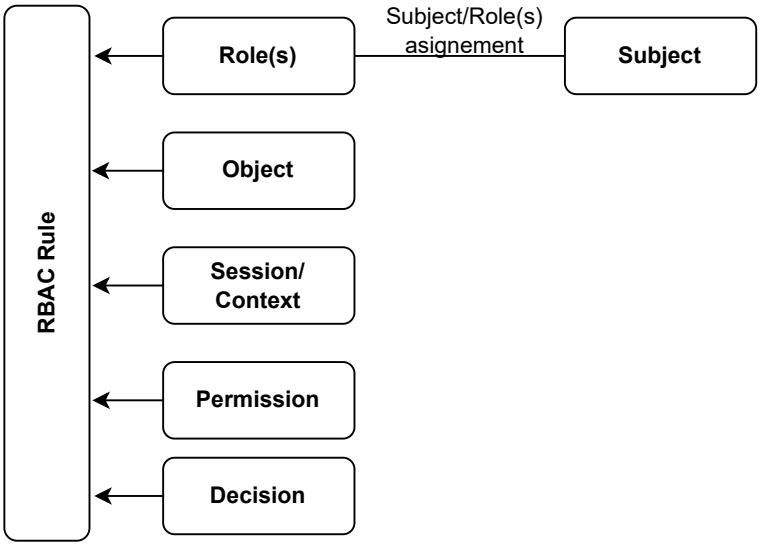


Figure 3.3 – RBAC policy specification model

of creating and maintaining roles is more time consuming than assigning permissions individually.

The following definition provides a formal modeling of an RBAC rule.

Definition 3.1 (RBAC Rule) *Formally an RBAC access control rule r is modeled as a quadruplet*

$$r = \langle \hat{R}, O, C, P, D \rangle$$

where $\hat{R}$, O, C, P, D represent respectively a role, an object, a context describing the condition(s) under which the rule can be fired, a permission representing a possible action on the object, and a decision stating whether the r is an authorization or a prohibition rule.

Attribute-Based Access Control(ABAC)

The ABAC model has been introduced in [PFMP04, YT05] to ensure more flexibility in the specification and the enforcement of access control requirements.

As illutrated in Figure 3.4, ABAC is a model that evaluates attributes assigned to subjects, objects, permission, and environment to determine access. ABAC as a form of logical access control became prominent in the past decade, having evolved from simple access control lists and role-based access control (RBAC) [11]. ABAC draws on

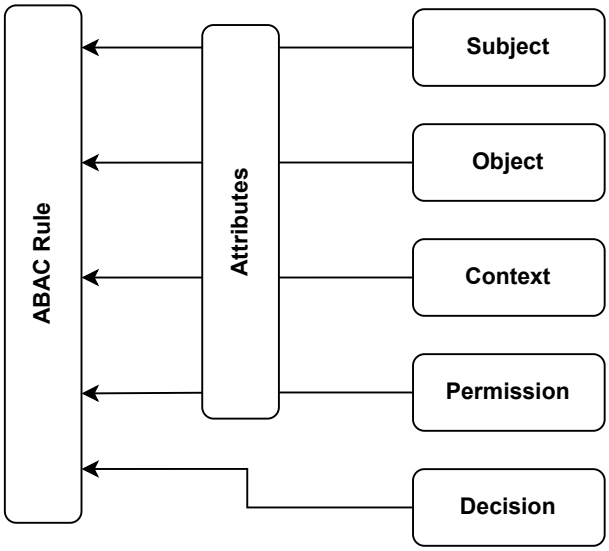


Figure 3.4 – ABAC policy specification model

a set of attributes (ex: user characteristics, environmental characteristics, resource attributes). ABAC supports Boolean logic, meaning that the attribution of access control can be dependent on a set of conditions (contrary to RBAC). This provides enhanced granularity and flexibility over previous models, including time-dependent access controls. On the downside, ABAC is complex to design and implement. The model's focus on attributes also makes it hard to gauge the permissions available to specific users before all attributes and rules are in place.

The following definition provides a formal modeling of an ABAC rule.

Definition 3.2 (ABAC Rule) *Let us denote by $\mathcal{A}$ the set of attributes that are to be used in the system to model the access control requirement. An ABAC access control rule r is modeled as a quadruplet*

$$r = \langle S = A^S, O = A^O, C = A^C, P = A^P, D \rangle$$

where S, O, C, P, and D represents respectively any subject of system to which the set of attributes $A^S \subseteq \mathcal{A}$, any object of system to which the set of attributes $A^O \subseteq \mathcal{A}$, a context describing the set of environment attributes $A^C \subseteq \mathcal{A}$ under which the rule can be fired, a permission modeled using a set of access attributes $A^P \subseteq \mathcal{A}$,and a decision stating whether the r is an authorization or a prohibition rule.

Organization-Based Access Control (ORBAC)

The ORBAC model [KBB+03] was first proposed to overcome the limitations of existing access control models such as RBAC. As shown in Figure 3.5, this model introduces a level of abstraction in which subjects are abstracted into roles, actions are abstracted into activities and objects are abstracted into views. Each security policy is defined by and for an organization and the high-level policy specification is parameterized by the organization by assigning to each subject a role, to each concrete action (i.e., read, write, delete, etc.) an activity, and to each object a view.

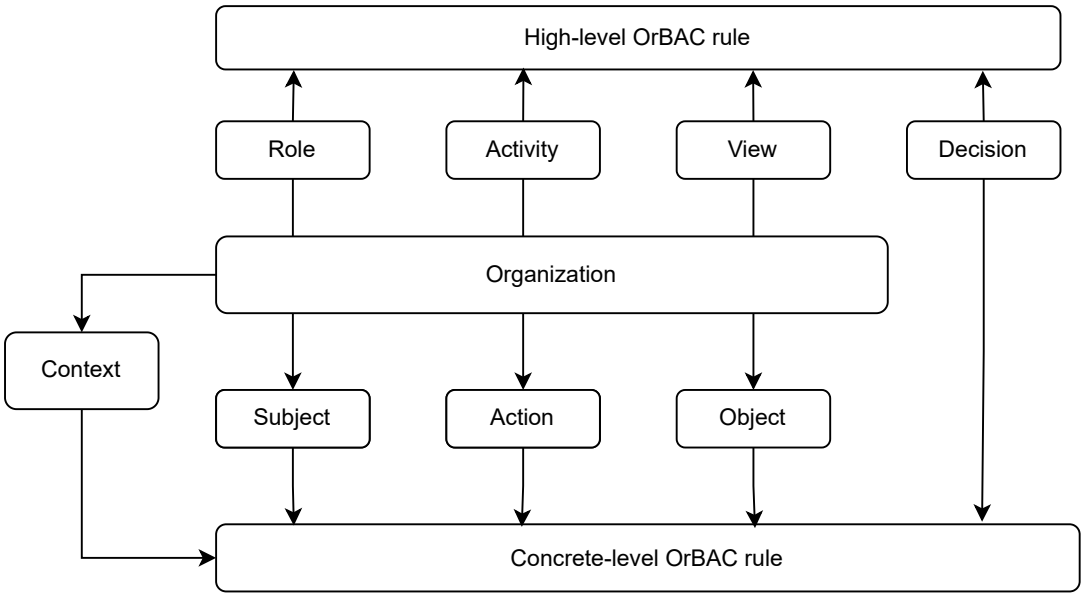


Figure 3.5 – An overview of the ORBAC model

ORBAC is still a dynamic model as context is taken into account as following:

- Permission means that given an organization, a role is authorized to perform an activity on a view if the context is active in the system.
- Prohibition states that a role is not authorized to perform some activity on some view when a given context is true.

ORBAC defines a formal refinement method allowing to transform abstract-level access control requirements to concrete level ones following the associations between a subject and a role, an action and an activity, and an object and a view.

Using the following definition, we provide a formal modeling of an ORBAC rule.

Definition 3.3 (ORBAC Rule [AEKEBB+03]) *An ORBAC access control rule r is modeled as a quadruplet*

$$r = \langle \texttt{Org}, \hat{R}, \hat{A}, \hat{V}, C, D \rangle$$

where `Org`*, $\hat{R}$, $\hat{A}$, $\hat{V}$, C, and D represents respectively the organization over which the rule is going to be enforced, a role, an activity, a view, a context, and a decision.*

3.3 The proposed model

As we have seen in the previous section, each of the aforementioned models specifies access control requirements in a different way. Our objective is then to define a new sufficiently generic access control specification model that can be used to express requirements specified using any of the aforementioned models.

In the rest of this section, we first present the adversary model we are going to consider in Section 3.3.1. Then, we propose a specification of a security policy to be deployed in Section 3.3.2. In section 3.3.5, the proposed model defines what is an access query and how it can be evaluated with regards to the policy to be enforced. Subsequently, in Section 3.3.6, we propose a provably correct policy refinement method. Finally, in Section 3.3.7, we focus on the specification of the NFV service requirements.

Afterwards, since DTE has made its evidence for the enforcement of access control policies at concrete level in operating system, a method for refining an access control policy specified using our policy specification model towards a concrete-level DTE specification is proposed in Section 3.3.4. The proposed transformation allows us to benefit from the advantages of DTE. In particular, it allows entities having the same access requirements to be collected into domains and types which allows to find an appropriate balance between security and policy deployment complexity. We prove that the proposed transformation method is correct and we show how the DTE policy is enforced.

3.3.1 Adversary Model

To understand the scope of the problem and evaluate the risks, we have to develop an adversary model. The adversary model considered in this work is composed of an attacker, an NFV infrastructure hosting the multiple VNFs that compose the network

services to be deployed, and an access control engine that is used to deploy the access control policy. In our model, the objective is to allow users including the likely attacker to perform operations that a normal NFV infrastructure user can do. It means that the attacker can generate and modify a flow attack to try to compromise a VNF. We suppose that the adversary will be able to interact with the VNFs composing the deployed network service but he cannot control or modify the behavior of the access control engine as well as the VNF that will be used to enforce the access control policy. This latter is supposed to be hardened i.e., keeping the operating system up to date, minimizing the installed packages to minimize vulnerabilities, enable and correctly configure a firewall, etc.

3.3.2 Security Policy specification

One of the properties that we want to ensure in our approach, is the high expressiveness. That is, we aim to define an access control policy specification model that can express requirements that are specified using any of the three well known access control models we aforementioned in Section 3.2.3. Hence, in the following, we define our property-based access control specification model (Definition 3.4), then we show how access control requirements specified using RABC, ABAC, or ORBAC can be translated to our property-based specification model.

Definition 3.4 (Access Control Policy) *An access control policy SP is composed of a set of access control rules* $\{r_1, \cdots, r_i\}$*. Each rule* r_i *comprises:*

- *A subject* S_i *that represents one or many entities that want to access the object. These entities are characterized by a set of properties* $\mathcal{P}_i^S = \{p_1^s, \cdots, p_n^s\}$*.*
- *An action* A_i *that represents the operation that is going to be performed by* S_i *on* O_i*. Each action is characterized by a set of properties* $\mathcal{P}_i^A = \{p_1^a, \cdots, p_l^a\}$*.*
- *An object* O_i *represents one or many resources over which the action* A_i *is going to be performed.* O_i *are characterized using two types of properties: (1) a set of properties* $\mathcal{P}_i^E$ *that characterizes the entities (e.g., VNF) and (2) a set of properties* $\mathcal{P}_i^R$ *that characterizes the resources inside those entities and over which the action* A_i *will be performed.*
- *A context* C_i *under which the rule can be invoked.*

- *A decision D_i indicating whether it is a permission or denial rule.*

In our model, each rule r_i in the security policy will be represented as follows:

$$r_i = \langle S_i := \mathcal{P}_i^S, A_i := \mathcal{P}_i^A, O_i := \{\mathcal{P}_i^E, \mathcal{P}_i^R\}, C_i, D_i\rangle$$

where $X := \mathcal{P}$ is used to denote the elements (e.g., subjects, objects, actions) that is characterized by the set of properties $\mathcal{P}$.

We note that the properties that characterize the entities representing the subject S_i (resp. the object O_i) may include: attributes of S_i (resp. O_i) in the considered system (e.g., the IP address of a network to which the subject belongs, etc.), functional attributes representing the provided functionalities (e.g., routing, deep packet inspection, firewalling, etc.), and security attributes that represent the security properties that is associated to S_i (resp. O_i) (e.g., the security level, trust level, etc.). In the case of NFV service, these properties can be retrieved from the VNF descriptors that compose the service to be deployed.

In the sequel, we will use **property-based specification model** to refer to our policy specification model defined in Definition 3.4.

Example 3.2 *To illustrate the security policy specification, let's take the example of a rule r_1 saying that any VNF providing web client functionalities and having a high security level can read the content of any web page on a VNF providing a web server functionality and having a high security level if the client is using https It can be specified using the following notation:*

$$\langle S := \{func : web\_client, sec\_level : high\}, A := \{Action : read, proto = https\}, O := \{\mathcal{P}^E = \{func : web\_server, sec\_level : high\}, \mathcal{P}^R = \{file\_name : any\}\}, C = \{between\ 8am\ and\ 8pm\}, D = allow\rangle$$

An access control policy is used to check all access requests that will be made in the system in which the policy is deployed. In the following definition, we define what an access request is, what it consists of, and to what extent it will be accepted or denied by the access control policy.

Definition 3.5 (Access Query) *An access query $\mathcal{AQ}$ is represented by the quadruplet $\langle S^q, O^q, A^q, C^q\rangle$ where S^q represents the subject performing the query, O^q the object*

over which the query is performed, A^q the action performed by the query, and C^q the request context under which the query is performed. Given a closed [1] *policy $\mathcal{SP}$, $\mathcal{AQ}$ is allowed by $\mathcal{SP}$ if and only if the following condition holds:*

(i) $\exists r_i \in \mathcal{SP}$ *such that* $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C^q *satisfies* C_i, *and* $D_i =$ *allow.*

$\mathcal{AQ}$ is denied by $\mathcal{SP}$ if and only if one of the following conditions hold:

(ii) $\nexists r_i \in \mathcal{SP}$ *such that* $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C^q *satisfies* C_i, *and* $D_i =$ *allow.*

(iii) $\exists r_i \in \mathcal{SP}$ *such that* $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C^q *satisfies* C_i, *and* $D_i =$ *deny.*

The conditions used in the previous definition can be informally described as follows:

- Condition *(i)* states that an access query $\mathcal{AQ}$ is allowed by the security policy $\mathcal{SP}$ if and only if there exists an authorization ($D_i = allow$) rule r_i in $\mathcal{SP}$ such that $\mathcal{AQ}$ satisfies r_i.
- Condition *(ii)* states that an access query $\mathcal{AQ}$ is denied by the security policy $\mathcal{SP}$ if there does not exist an authorization rule r_i in $\mathcal{SP}$ such $\mathcal{AQ}$ satisfies r_i.
- Condition *(iii)* states that an access query $\mathcal{AQ}$ is denied by the security policy $\mathcal{SP}$ if there exists a prohibition ($D_i = Deny$) rule r_i in $\mathcal{SP}$ such $\mathcal{AQ}$ satisfies r_i.

3.3.3 Policy translation

We design our access control specification model presented in Definition 3.3 to be sufficiently expressive so that it can be used to specify any access control policy expressed using RBAC, ABAC, and ORBAC. In the following, we propose a translation method for each of the considered access control model. Then we prove their correctness.

Definition 3.6 (Policy translation correctness) *Let us consider two access control specification model $\mathcal{M}_i$ and $\mathcal{M}_j$ and a rule r^i specified using $\mathcal{M}_i$. Let us denote by r^j the translation of r^i in $\mathcal{M}_j$. The translation is correct if and only if there exist no access query $\mathcal{AQ}$ such that none of the following conditions hold:*

(i) $\mathcal{AQ}$ satisfies r^i but not r^j.

(ii) $\mathcal{AQ}$ satisfies r^j but not r^i

1. only accesses that are explicitly authorized should be allowed, all other accesses will be denied.

RBAC rule translation

The translation of an RBAC rule to our property-based specification model is depicted in Figure 3.6.

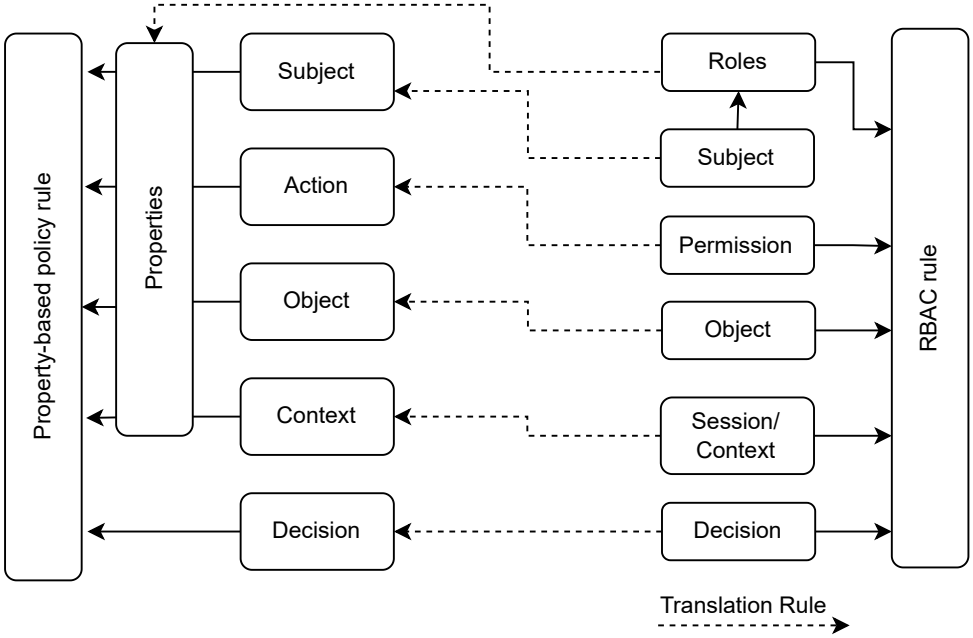


Figure 3.6 – Translation of an RBAC rule to a property-based rule

The RBAC notions subject, object, context, and decision are straightforwardly translated to the same notion used in our property-based model. The notion of permission is translated to an action that can be performed over the object. Finally, The notion of role can be seen in our model as a specific property that can be associated to a subject. We formalize the previous translation using the following definition.

Definition 3.7 *An RBAC rule* $r = \langle \hat{R}, O, C, P, D \rangle$ *is translated to a property-based specification rule* r^* *where*

$$r^* = \langle S^* := \{role : \hat{R}\}, A^* = P, O^* = O, C^* = C, D^* = D \rangle$$

Proposition 3.3.1

The RBAC translation method defined in Definition 3.7 is correct.

Proof: Proof is by contradiction. According to Definition 3.6, a translation is correct if and only if there exists no access query $\mathcal{AQ}$ such that one of the following conditions hold:

1. $\mathcal{AQ}$ satisfies r but not r^*
2. $\mathcal{AQ}$ satisfies r^* but not r

So let us suppose that there exists $\mathcal{AQ} = \langle S^q, O^q, A^q, C^q \rangle$ such that one of the previous conditions hold. Two cases should be considered.

- Case 1: condition (1) holds. As $\mathcal{AQ}$ satisfies r, then we can deduce (d1) that S^q has the role $\hat{R}$, $O^q = O$, $A^q = P$, and C^q satisfies C. Now, considering the fact that $\mathcal{AQ}$ does not satisfy r^*, then we can deduce that S^q does not have the role $\hat{R}$, $A^q \neq A^*$, $O^q \neq O^*$, or C^q does not satisfy C^*, which contradicts (d1).
- Case 2: condition (2) holds. As $\mathcal{AQ}$ satisfies r^*, then we can deduce (d2) that S^q has the role $\hat{R}$, $O^q = O^*$, $A^q = A^*$, and C^q satisfies C^*. Now, considering the fact that $\mathcal{AQ}$ does not satisfy r, then we can deduce that S^q does not have the role $\hat{R}$, $A^q \neq P$, $O^q \neq O$, or C^q does not satisfy C, which contradicts (d2).

□

ABAC rule translation

An ABAC rule can be translated to our property-based specification model as illustrated in Figure 3.7. Similarly to the RBAC rule translation method, the ABAC notions subject, object, context, and decision are straightforwardly translated to the same notion used in our property-based model. The permission notion is translated to an action. In addition, the attributes used in the ABAC model to model the subject, object, permission, and context, can be translated in our model to properties that characterize a subject, an object, a context, or an action.

The translation of ABAC rules to our property-based specification rules is formalized as follows.

Definition 3.8 *An ABAC rule* $r = \langle S = A^S, O = A^O, C = A^C, P = A^P, D \rangle$ *is translated to a property-based specification rule* r^* *such that*

$$r^* = \langle S^* := \mathcal{P}^S, A^* := \mathcal{P}^P, O^* := \mathcal{P}^O, C^* := \mathcal{P}^C, D \rangle$$

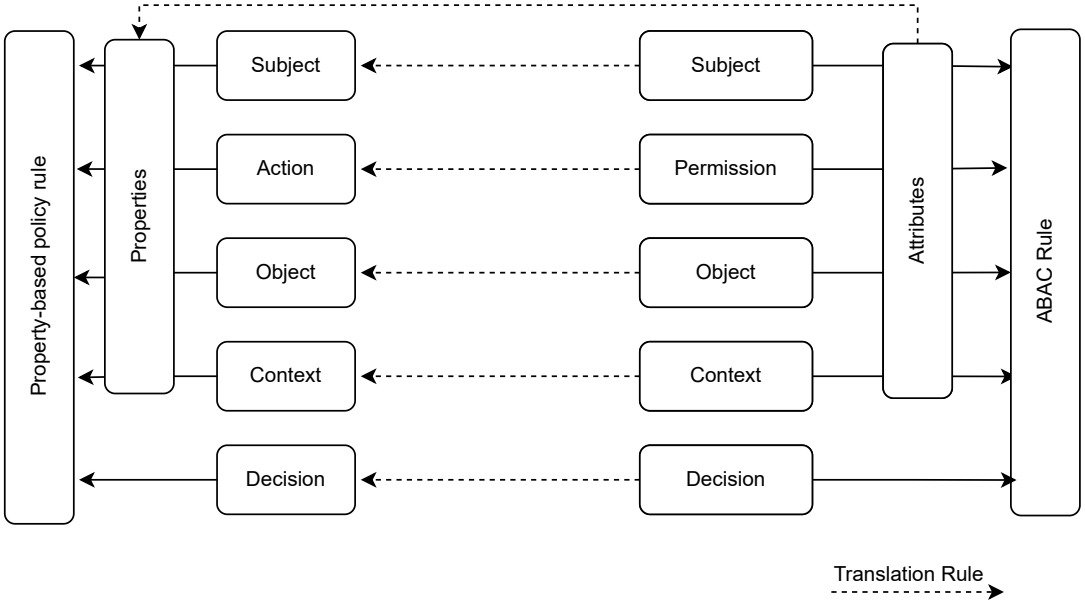


Figure 3.7 – Translation of an ABAC rule to a property-based rule

where $\mathcal{P}^X$ are the set of properties that translates the set of attributes A^X.

Proposition 3.3.2

The translation method of ABAC rules to a property-based specification rules defined in Definition 3.8 is correct.

Proof: By following the same reasoning method as in the proof of Proposition 3.3.1, we can straightforwardly prove the previous proposition. Proof is by contradiction. According to Definition 3.6, a translation is correct if and only if there exists no access query $\mathcal{AQ}$ such that one of the following conditions hold:

1. $\mathcal{AQ}$ satisfies r but not r^*
2. $\mathcal{AQ}$ satisfies r^* but not r

So let us suppose that there exist $\mathcal{AQ} = \langle S^q, O^q, A^q, C^q \rangle$ such that one of the previous conditions hold. Two cases should be considered.

- Case 1: condition (1) holds. As $\mathcal{AQ}$ satisfies r, then we can deduce (d1) that S^q has the role $\hat{R}$, $O^q = O$, $A^q = P$, and C^q satisfies C. Now, considering the fact

that $\mathcal{AQ}$ does not satisfy r^*, then we can deduce that S^q does not have the role $\hat{R}$, $A^q \neq A^*$, $O^q \neq O^*$, or C^q does not satisfy C^*, which contradicts (d1).

- Case 2: condition (2) holds. As $\mathcal{AQ}$ satisfies r^*, then we can deduce (d2) that S^q has the role $\hat{R}$, $O^q = O^*$, $A^q = A^*$, and C^q satisfies C^*. Now, considering the fact that $\mathcal{AQ}$ does not satisfy r, then we can deduce that S^q does not have the role $\hat{R}$, $A^q \neq P$, $O^q \neq O$, or C^q does not satisfy C, which contradicts (d2).

□

ORBAC rule translation

ORBAC rules can be translated to our property based specification model as illustrated in Figure 3.8.

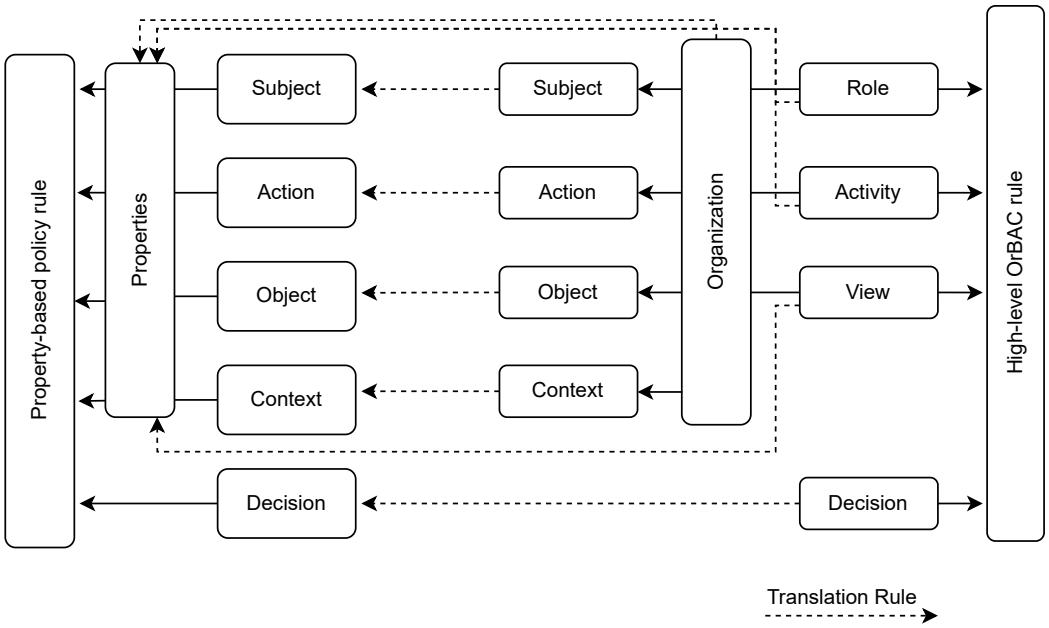


Figure 3.8 – Translation of an OrBAC rule to a property-based rule

To be able to translate high-level ORBAC rule to our specification model, we consider the role, activity, view notions of the ORBAC model as properties that can be assigned respectively to a subject, an action, and an object in our specification model. The ORBAC's organization notion is translated to a property that can be associated to any subject, object, action, and context. Finally, the ORBAC concrete-level notions subject, object, action, are straightforwardly translated to the same notions used in our

property-based model. The translation of ORBAC rules to our property-based specification rules is formalized as follows.

Definition 3.9 *An ORBAC rule* $r = \langle \texttt{Org}, \hat{R}, \hat{A}, \hat{V}, C, D \rangle$ *is translated to a property-based specification rule* r^* *such that*

$$r^* = \langle S^* := \{role : \hat{R}, org : \hat{\texttt{Org}}\}, A^* := \{activity : \hat{A}, org : \hat{\texttt{Org}}\}, O^* := \{view = \hat{V}, org : \hat{\texttt{Org}}\}, C^* = C, D^* = D \rangle$$

Remark 3.3.1

In the previous Definition, the property *org* is only involved in the specification of the subject, action, and object of the rule r^*. This choice is motivated by the fact that the association between the high-level concepts such as role, activity and view with low level concepts such as subject, object, and action is to be decided by the organization, and will affect only the subjects, objects and actions related to the organization. This modeling allows us also to be able to specify access control policies defined by several organizations.

Proposition 3.3.3

The translation method of ORBAC rules to a property-based specification rules defined in Definition 3.9 is correct.

Proof: We prove the previous proposition by contradiction. Similarly to the proof of Proposition 3.3.1, we suppose that there exists an access query $\mathcal{AQ} = \langle S^q, O^q, A^q, C^q \rangle$ such that the translation method defined in Definition 3.9 is not correct. So according to Definition 3.6, to cases should be considered:

- Case 1: $\mathcal{AQ}$ satisfies r but not r^*.
- Case 2: $\mathcal{AQ}$ satisfies r^* but not r.

For both previous cases, a contradiction is shown as follows.

Case 1: Since $\mathcal{AQ}$ satisfies r, we can deduce that (d1) the role $\hat{R}$ is assigned to S^q within the organization $\hat{\texttt{Org}}$, the object O^q belongs to the view $\hat{V}$ within $\hat{\texttt{Org}}$, the action

A^q is part of the activity $\hat{A}$ within $\hat{\texttt{Org}}$, and the context C^q satisfies C. In the other hand, $\mathcal{AQ}$ does not satisfies r^* means that S^q does not have the role $\hat{R}$ in $\hat{\texttt{Org}}$, O^q does not belong to the view $\hat{V}$ within $\hat{\texttt{Org}}$, the action A^q is not part of the activity $\hat{A}$ within $\hat{\texttt{Org}}$, or C^q does not satisfy C, which contradicts (d1).

Case 2: Following the same reasoning as in the previous case, we can straightforwardly show a contradiction and conclude the proof. □

Remark 3.3.2

In this section, we propose an access control policy modeling language, then we formally proved through Propositions 3.3.1, 3.3.2, and 3.3.3 that it can be used to correctly express access control policies specified in RBAC, ORBAC and ABAC. Generally, our model can be used to specify any access control policy that is based on the properties related to the involved subjects, objects, and actions. Hence, we answer positively the research question RQ-5.1.

3.3.4 Policy refinement

In this section, we propose a method for refining a policy specified using our property-based policy specification model (Definition 3.4) towards a concrete-level specification. Then, we formally prove that the refinement method we propose is correct.

Remark 3.3.3

In our approach, we choose DTE as a concrete enforcement model as it offers several advantages. First, as mentioned in Section 3.2.2, DTE allows to manage both access Control by controlling access from domains to types and flow Control by controlling transitions from domains to other domains. Second, the DTE model has been widely adopted for enhancing the security of Linux operating systems through the Security-Enhanced Linux (SELinux) [Cla00] module implementing the DTE model. Third, the impact of the enforcement of a DTE policy in Linux operating system on the different operations (e.g., file read, file write, file copy, etc) is very low (less than 10 μs) [ZLJ21]. Hence, by using DTE as a concrete enforcement model in NFV service, we expect to have a quite low latency on the NFV quality of service due to the evaluation of the DTE specification.

Definition 3.10 *Given a security policy $\mathcal{SP}$ composed of n rules $r_1, \cdots, r_n$. $\mathcal{SP}$ is transformed to a DTE policy by performing, for each rule $r_i \in \mathcal{SP}$ the following steps that are also depicted in Figure 3.9:*

- ***Step 1:*** *If there exist no $j < i$ such that $\mathcal{P}_i^S = \mathcal{P}_j^S$, define the domain $s\_\mathcal{P}_i^S\_d$ which will contain all entities of the considered system (i.e., the network service to be deployed) that have the set of properties $\mathcal{P}_i^S$ used to characterize the subject S_i. Otherwise, use the same domain $s\_\mathcal{P}_j^S\_d$ (i.e., $s\_\mathcal{P}_i^S\_d = s\_\mathcal{P}_j^S\_d$) defined for the subject S_j of the rule r_j.*

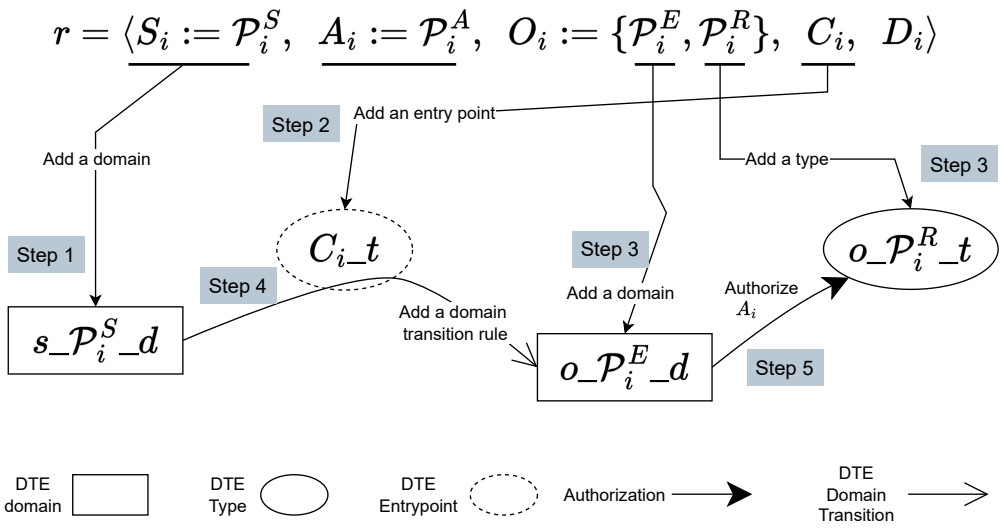


Figure 3.9 – The refinement of a property-based rule towards a DTE specification

- ***Step 2:*** *If there exist no $k < i$ such that $C_i = C_k$, define new type c_i_t. Otherwise, use the same type c_k_t defined for the context of the rule r_k $(i.e., c_i\_t = c_k\_t)$. Note that the type c_i_t will be considered as an entry point allowing any access query associated to c_i_t to transit through the entry point.*

- ***Step 3:*** *If there exist no $l < i$ such that $\mathcal{P}_i^R = \mathcal{P}_l^R$ define new type $o\_\mathcal{P}_i^R\_t$ which will be associated to all resources of the considered system that have the set of properties $\mathcal{P}_i^R$. Otherwise, use the same type $o\_\mathcal{P}_l^R\_t$ (i.e., $o\_\mathcal{P}_l^R\_t = o\_\mathcal{P}_i^R\_t$). In addition, if there exist no $l' < i$ such that $\mathcal{P}_i^E = \mathcal{P}_l^E$, define new domain $o\_\mathcal{P}_i^E\_d$ that will contain all entities of the considered system that have the set of properties*

$\mathcal{P}_i^E$. *Otherwise, use the same domain* $o\_\mathcal{P}_{l'}^E\_d$ *(i.e.,* $o\_\mathcal{P}_{l'}^E\_d = o\_\mathcal{P}_i^E\_d$*) defined for the object of the rule* r_l*.*

- ***Step 4:*** *When associated to the request context* C^q *of an access query* $\mathcal{AQ}$ *(i.e., the context of the rule* r_i *is satisfied by the context* C^q *of* $\mathcal{AQ}$*), allow the type* c_i_t *to be an entry point allowing to transit* $\mathcal{AQ}$ *from domain* $s\_\mathcal{P}_i^S\_d$ *to the domain* $o\_\mathcal{P}_i^O\_d$*.*

- ***Step 5:*** *Authorize access queries that transit from* $s\_\mathcal{P}_i^S\_d$ *to* $o\_\mathcal{P}_i^E\_d$ *to perform the actions* A_i *on any objects having the type* $o\_\mathcal{P}_i^R\_t$*.*

Finally, we denote by $\mathcal{C}$ *the set of couples of* DTE *types* c_i_t *and their respective contexts* C_i *created in step 2* $(\mathcal{C} = \{(c_i\_t, C_i)\})$*.*

Remark 3.3.4

We note here that only the authorization rules i.e., having an *allow* decision, are considered in the previous definition. This choice is due to the fact that DTE is using closed policies by default. Hence, there is no need to refine prohibition rule to a DTE specification. We emphasis here that the refinement strategy presented in Definition 3.10 can only be used with access control policies that does not contain conflictual rules i.e., rules that both allow and deny an access query. A method for dealing with such access control policies is proposed in Chapter 4.

In our model, when an access query is created by the system, the query inherits all the types associated with the subject S^q and belongs to the DTE domains of S^q. In addition, we suppose that the system associates types to C^q as follows. $\forall (c_i\_t, C_i) \in \mathcal{C}$: if C_i is satisfied in C^q, then associate the type c_i_t to the request context C^q of the access query.

Example 3.3 *This example illustrates the security policy transformation method we defined in Definition 3.10. Let us consider that we have a security policy* $\mathcal{SP}$ *that is composed of three rules* r_1, r_2 *and* r_3 *such that:*

- $r_1 = \langle S_1 := \{func : web\_server, sec\_level : high\}, A_1 = read, O_1 := \{\mathcal{P}_1^E = \{func : ftp\_server, sec\_level : high\}, \mathcal{P}_1^R = \{file\_name : any\}\}, C_1 = \{$*between 8am and 8pm*$\}, D_1 = allow\rangle$

- $r_2 = \langle S_2 := \{func : web\_server, sec\_level : high\}, A_2 = write, O_2 := \{\mathcal{P}_2^E = func : database\_server, sec\_level : low, \mathcal{P}_2^R = db\_name : service\_db\}, C_2 = \{$*between 8am and 8pm*$\}, D_2 = allow\rangle$

- $r_3 = \langle S_3 := \{func : web\_client, sec\_level : high\}, A_3 = access, O_3 := \{\mathcal{P}_3^E = func : ftp\_server, sec\_level : high, \mathcal{P}_3^R = file\_name : web\_config\}, C_3 = \{$*between 8am and 8pm*$\}, D_3 = allow\rangle$

According to Definition 3.10, the transformation of the policy $\mathcal{SP}$ to a DTE specification is illustrated using the schema in Figure 3.10.

Subjects $S_1, S_2,$ of rules r_1, r_2 are respectively represented in the transformation by the DTE domain $s\_web\_server\_high\_d$ while the subject S_3 of r_3 is represented by the domain $s\_web\_client\_high\_d$. The entities of objects O_1 and O_3 of r_1 and r_3 (described using the set of properties P_1^E and P_3^E) are represented in the DTE transformation by the DTE domain $o\_ftp\_server\_high\_d$ and the resources of the objects O_1 and O_3 (described using the set of properties P_1^R and P_3^R). In the case of O_2, the entities described using the set of properties P_2^E and the resources are described using the set of properties P_2^R.

The DTE domains $o\_ftp\_server\_high\_d$ and $o\_db\_server\_low\_d$ are respectively created by the transformation of the rules r_1 and r_3. After the transformation, $o\_ftp\_server\_high\_d$ contains the ftp server having the security level high while the domain $o\_db\_server\_low\_d$ contains the database servers having the security level low. Finally, c_t is a DTE type that will be associated to any access query satisfying the context C_1, C_2, and C_3 of the rules r_1, r_2, and r_3.

3.3.5 Access query evaluation

In this section, we focus on how the access query will be evaluated with respect to the DTE specification that are to be enforced. The pseudo-code in Algorithm 1 outlines the procedure used to authorize an access query. It takes as inputs the access query to be authorized, the sets $\mathcal{C}$ of types and their respective contexts created in the policy transformation process (Definition 3.10). It outputs a value (*allow_traffic* or *deny_traffic*) indicating whether or not the access query should be allowed by the DTE policy. The functions *get_domains*, *get_types*, *get_transition_src*, and *get_transition_dst* are used to retrieve respectively, the set of domains to which the subject of the access query

belongs, the set of types associated with the object of the access query, the source domain of the domain transition, and the destination domain of the transition. The function *assign_types* assigns to the access query $\mathcal{AQ}$ the types used in $\mathcal{C}$ if their respective context matches the context of $\mathcal{AQ}$. Finally, the function *check_permission* checks whether a DTE domain is allowed or denied to perform the actions in $\mathcal{A}^q$ on a given DTE type.

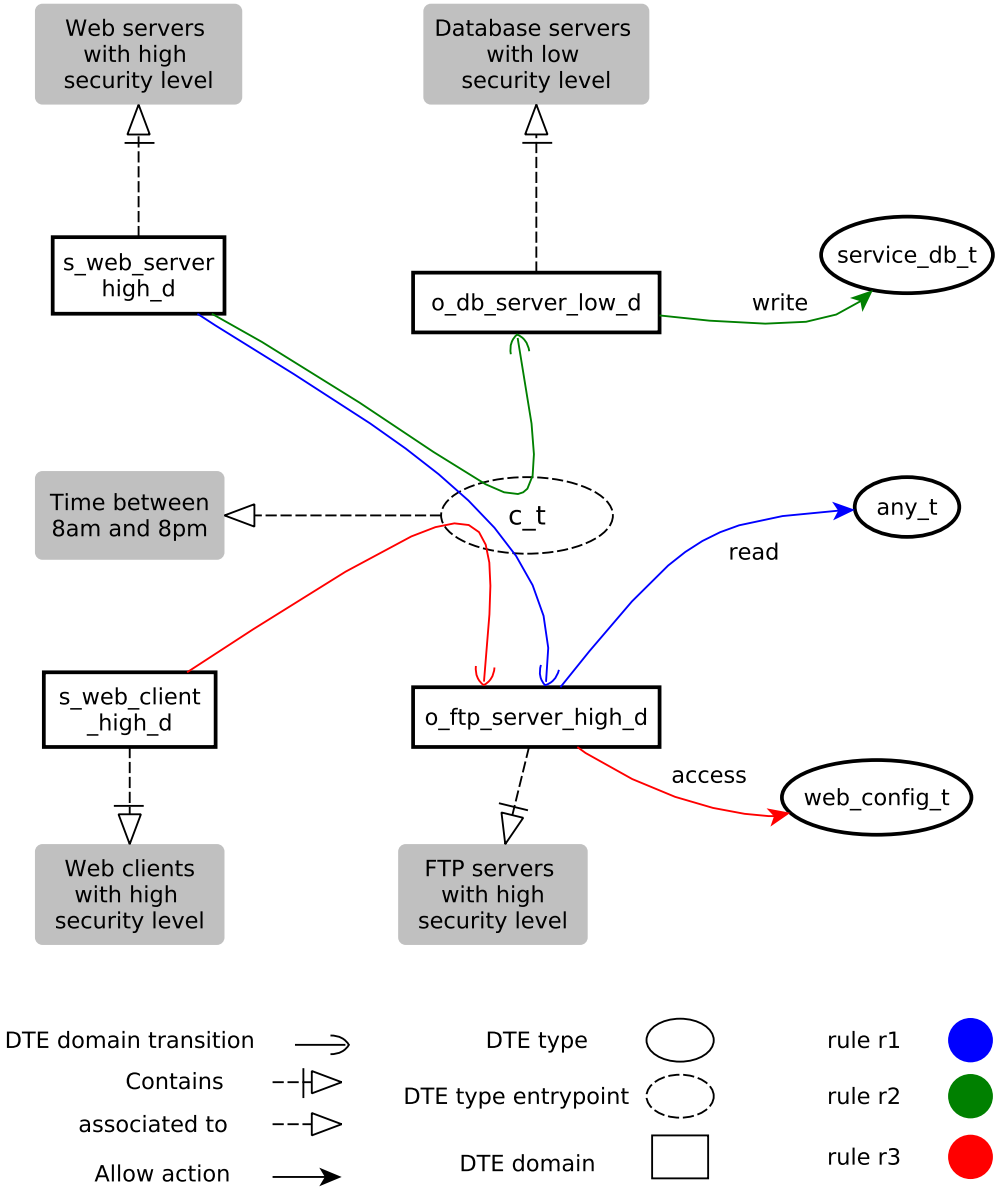


Figure 3.10 – Transformation of rules r_1, r_2, and r_3 to a DTE policy

The foreach loop (lines 5 to 8) is used to find all the possible DTE domains to which the access query can transit. Then, using the second foreach loop (lines 10 to 17), the procedure checks if there exists one of the domains to which the query can transit that allows to perform the action $\mathcal{A}^q$ on one of the types associated to the target object O^q.

Input: $\mathcal{AQ} : \langle S^q, O^q, \mathcal{A}^q, C^q \rangle$ /* the access query be authorized */
$\mathcal{C} = \{(c\_r_i\_t, C_i)\})$ /* Definition 3.10 (step 2) */

```
1  subject_domains = get_domains(S^q)
2  object_types = get_types(O^q)
3  AQ_types = assign_types(AQ, C)
4  possible_domains = ∅
5  foreach AQ_type ∈ AQ_types do
6      if AQ_type is not a DTE entrypoint then continue ;
7      if get_transition_src(AQ_type) ∉ subject_domains then continue ;
8      possible_domains = possible_domains ∪ get_transition_dst(AQ_type)
9  end
10 foreach type ∈ object_types do
11     foreach domain ∈ possible_domains do
12         if check_permission(domain, type, A^q) = allow then
13             transit_domain(AQ, domain);
14             return allow_query;
15         else if check_permission(domain, type, A^q) = deny then
16             return deny_traffic;
17     end
18 end
```

Algorithm 1: Access query authorization

Example 3.4 *To illustrate how an access query is going to be evaluated, let us consider the access control policy that has been refined in Example 3.3, and the following two access queries*

$\mathcal{AQ}_1 : \langle S_1^q = web\_client, O_1^q = ftp\_server :: web\_config\_t, A_1^q = read, C_1^q = \{$*time=12 am*$\}\rangle$

$\mathcal{AQ}_2 : \langle S_2^q = web\_client, O_2^q = db\_server, A_2^q = access, C_2^q = \{$*time=12 am*$\}\rangle$

to be evaluated and performed on the considered system. We suppose the following:

- *The* web_client *entity has high security level, so it belongs to* $s\_web\_client\_high\_d$ *domain.*

- *The ftp_server entity has high security level, so it belongs to $s\_ftp\_server\_high\_d$ domain.*

- *db_server entity has low security level and then belongs to $o\_db\_server\_low\_d$ domain.*

So, let us first focus on $\mathcal{AQ}_1$. According to the access query evaluation procedure presented in Algorithm 1, $\mathcal{AQ}_1$ will inherit the domain of its subject, so it will initially belongs to the domain $s\_web\_client\_high\_d$. Moreover, the context C_1^q of $\mathcal{AQ}_1$ satisfies the contexts C_1, C_2 and C_3 of the three rules r_1, r_2, and r_3. Hence, the type c_t will be associated to the query $\mathcal{AQ}_1$. According to the step 4 of our transformation method, the DTE type c_t will allow the access query $\mathcal{AQ}_1$ to transit from the domain $s\_web\_client\_high\_d$ to the domain $o\_ftp\_server\_high\_d$. Furthermore, according to the transformation shown in Figure 3.10, the action access is authorized to be performed by access queries belonging to the domain $o\_ftp\_server\_high\_d$ over objects associated to the DTE type $o\_ftp\_server\_high\_t$. So, we conclude that the query $\mathcal{AQ}$ is to be authorized by the DTE specification of $\mathcal{SP}$.

Let us now focus on the query $\mathcal{AQ}_2$. The latter will be associate initially to the domain of its subject $s\_web\_client\_high\_d$. As we can see in Figure 3.10, the only possible domain to which $\mathcal{AQ}_2$ can transit is $o\_ftp\_server\_high\_d$, and no action is possible from $o\_ftp\_server\_high\_d$ on resources associated to the DTE object $service\_db\_t$. Hence, $\mathcal{AQ}_2$ will be denied by the DTE specification of $\mathcal{SP}$.

3.3.6 Policy refinement correctness

Since the policy is being refined from our high-level property-based specification model to a concrete DTE specification, we must be sure that the transformation is correct in the sense that the policy that is going to be deployed in the DTE level is equivalent to the high-level policy that has been specified.

A security policy transformation method is correct if, for any access query, no rule in the transformed security policy is violated when the transformation resulting policy is deployed. This is formalized using the following definition.

Definition 3.11 *Given a closed security policy $\mathcal{SP} = \{r_1, \cdots, r_n\}$ and its corresponding DTE transformation $\mathcal{SP}_{DTE}$ (as described in Definition 4.3). The transformation*

from $\mathcal{SP}$ to $\mathcal{SP}_{DTE}$ is correct if and only if for any access query $\mathcal{AQ}$: if $\mathcal{AQ}$ is allowed (resp. denied) by $\mathcal{SP}$, then it is allowed (resp. denied) by $\mathcal{SP}_{DTE}$.

Theorem 3.3.1

The policy transformation method proposed in Definition 3.10 is correct.

Proof: We prove the previous theorem by contradiction. Let us denote by $\mathcal{SP}$ the transformed policy and $\mathcal{SP}_{DTE}$ the transformation resulting policy. According to Definition 3.11, the policy transformation method is not correct if one of the following cases hold:

- **case 1:** There exists an access query $\mathcal{AQ}$ such that it is allowed by $\mathcal{SP}$ and denied by $\mathcal{SP}_{DTE}$.
- **case 2:** There exists an access query $\mathcal{AQ}$ such that it is denied by $\mathcal{SP}$ and allowed by $\mathcal{SP}_{DTE}$.

For both cases, a contradiction is shown in the following.

Case 1: Formally, this case implies that $\exists r_i \in \mathcal{SP}, \exists \mathcal{AQ}$ such that: $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C_i is satisfied in C^q, and $D_i =$ allow. $S^q \in S_i$ means that S^q will belong to the same domain as S_i ($s\_\mathcal{P}^{S_i}\_d$) and that the query itself will belong to $s\_\mathcal{P}^{S_i}\_d$. According to the step 3 of our policy transformation method (Definition 4.3), $O^q \in O_i$ implies that the object O^q will have the type $o\_\mathcal{P}^{R_i}\_t$. In addition, according to the query initialization rules, C_i is satisfied in C^q means that the type c_i_t will be assigned to C^q. Then, according the step 4 of Definition 4.3, when executed, $\mathcal{AQ}$ will transit from the domain $s\_\mathcal{P}^{S_i}\_d$ to the domain $o\_\mathcal{P}^{E_i}\_d$. Subsequently, and according to the step 5 of Definition 4.3, since $\mathcal{AQ}$ transited to $o\_\mathcal{P}^{E_i}\_d$, it will have the permission to perform the set of actions A_i on all entities having the type $o\_\mathcal{P}^{R_i}\_t$. Finally, since $O^q \in O_i$ and $A^q \in A_i$, then $\mathcal{AQ}$ will have the permission to perform the action A^q on the object O^q which contradicts the hypothesis of the case 1.

Case 2: This case happens if one of the following conditions hold:

case 2.1: Given the access query $\mathcal{AQ}$, there exists no rule in the policy $\mathcal{SP}$ that allow $\mathcal{AQ}$. Formally, $\nexists r_i \in \mathcal{SP}$ such that $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C^q satisfies C_i, and $D_i =$ allow. Let us suppose that the $\mathcal{AQ}$ is allowed by $\mathcal{SP}_{DTE}$. According to the transformation method, action permission is only specified in step 5 of Definition 4.3. This step means that if $\mathcal{AQ}$ is allowed by $\mathcal{SP}_{DTE}$, then there exist a domain $s\_\mathcal{P}^{S_i}\_d$ and a type $o\_\mathcal{P}^{R_i}\_t$ such that $\mathcal{AQ}$ belongs to $s\_\mathcal{P}^{S_i}\_d$ and O^q has the type $o\_\mathcal{P}^{R_i}\_t$. This means that there exists $r_i \in \mathcal{SP}$ such that $A_q \in A_i$ and $O^q \in O_i$. In addition, according to step 3 of Definition 4.3, $o\_\mathcal{P}^{R_i}\_d$ $(i \in [1, n])$ does not contain any access query when created. These domains are only accessible for access queries thought the transformation rule defined in step 4 of Definition 4.3. Since we already showed that $\mathcal{AQ}$ belongs to $s\_\mathcal{P}^{S_i}\_d$, then there exists an entrypoint type c_i_t that allows $\mathcal{AQ}$ to transit to the domain $o\_\mathcal{P}^{E_i}\_d$ which allow us to deduce that C_i is satisfied in C^q and that both S_i and S^q belong to the same domain $s\_\mathcal{P}^{S_i}\_d$ (since $S^q \in S_i$). Then, we deduce that $\exists r_i \in \mathcal{SP}$ such that $S^q \in S_i$, $O^q \in O_i$, $A^q \in A_i$, C_i satisfied in C^q and $D_i = allow$ which contradicts the case 2.1. □

case 2.2: This case implies that given the access query $\mathcal{AQ}$, in one hand $\exists r_i \in \mathcal{SP}, \exists \mathcal{AQ}$ such that: $S^q \in S_i, O^q \in O_i, A^q \in A_i$, C_i is satisfied in C^q, and $D_i =$ deny and in the other hand $\mathcal{AQ}$ is allowed by $\mathcal{SP}_{DTE}$. $S^q \in S_i$ means that S^q will belong to the same domain as S_i ($s\_\mathcal{P}^{S_i}\_d$) and that the query itself will belongs to $s\_\mathcal{P}^{S_i}\_d$, since $\mathcal{AQ}$ inherit the domain of its subject then $\mathcal{AQ}$ belongs also to $s\_\mathcal{P}^{S_i}\_d$. Since the rule r_i is transformed using our transformation method, then there exists the type c_i_t that represents an entrypoint to the domain $o\_\mathcal{P}^{E_i}\_d$. Since C_i is satisfied in C^q, the type c_i_t will be associated to the C^q of $\mathcal{AQ}$, as a result, when executed, $\mathcal{AQ}$ will transit from $s\_\mathcal{P}^{S_i}\_d$ to $o\_\mathcal{P}^{E_i}\_d$. However, based on the transformation of r_i, the domain $o\_\mathcal{P}^{E_i}\_d$ will be denied to perform the action A_i on the type $o\_\mathcal{P}^{R_i}\_t$. Finally, since $O^q \in O_i$ and $A^q \in A_i$ then $\mathcal{AQ}$ will be denied by the $\mathcal{SP}_{DTE}$ which contradicts the case 2.2.

Remark 3.3.5

As we have seen in this section, we have proposed a refinement transformation method that allows to refine high level policies towards DTE specification, we show how we can evaluate access queries against a DTE specification and we prove the correctness of our refinement method. Hence, with the previous we answered the research question RQ-5.2.

3.3.7 Service requirements specification

The goal of our model is to allow the enforcement of access control policies in NFV services to provide their intended function. To meet the previous objective, the traffic that needs to flow through the target NFV service should be authorized by the access control policy to be enforced. In this section, we focus on the specification of the NFV service requirements and how they can be translated to property-based access control rules.

An NFV service $\mathcal{S}$ is composed of a set of VNFs $\{vnf_1, \cdots, vnf_n\}$ and a set of forwarding graphs $\{fg_1, \cdots, fg_m\}$. Each forwarding graph fg_i is composed of a set of forwarding paths $\{fp_1, \cdots, fp_d\}$, each fp_i can be represented using the following couple

$$\langle \langle vnf_1^i, vnf_2^i, \cdots, vnf_{n_i}^i \rangle, fp\_m_i \rangle$$

where vnf_1^i is the VNF that is forwarding the traffic, vnf_n^i is the VNF to which the traffic is forwarded, and fp_m_i is the match policy that will be used to distinguish which traffic should traverse the path.

In our model, a traffic $\mathcal{T}$ is used to represent each exchange between two consecutive VNFs in the considered forwarding path. It is modeled as the quadruplet

$$\langle vnf\_src, vnf\_dst, t\_context, t\_content \rangle$$

where vnf_src, vnf_dst, $t\_context$, and $t\_content$ represent respectively, the VNF that is sending the traffic, the VNF destination of the traffic, the context and the content of the traffic. Formally, a forwarding path $fp = \langle \langle vnf_1, vnf_2, \cdots, vnf_n \rangle, fp\_m \rangle$ is represented using $n-1$ traffics $\mathcal{T}_i = \langle vnf_i, vnf_{i+1}, fp_m, t\_content \rangle$, $1 \leq i < n$.

It is worth highlighting that the action involved in the security policy to be deployed can be implemented in the content of a traffic according to the used application protocol. For example, the "write" action can be implemented according to the protocol that is used. If the FTP protocol is used, a traffic containing the "post" ftp command implements the action "write" used in the security policy. Based on the previous observation, a traffic can be modeled as an access query as defined in the following.

Definition 3.12 *A traffic $\mathcal{T} = \langle vnf\_src, vnf\_dst, t\_context, t\_content \rangle$ will be modeled as an access query $\mathcal{AQ} = \langle S^q, O^q, \mathcal{A}^q, C^q \rangle$ where vnf_src equals to S^q, vnf_dst equals to O^q, $\mathcal{A}^q$ are the actions that can be implemented by the traffic content $t\_content$, and*

$C^q = t\_context.$

To ensure a proper functioning of the NFV service to be deployed, the traffics that represent each forwarding path should be allowed to flow according to the latter. To meet the previous objective, for each traffic $\mathcal{T}_i = \langle vnf_i, vnf_{i+1}, t_i\_context, t_i\_content \rangle$ that is modeled as the access query $\mathcal{AQ}_i = \langle vnf_i, vnf_{i+1}, \mathcal{A}_i^q, C_i^q \rangle$, we define the following policy rule:

$$r_{\mathcal{T}_i} = \langle S = vnf_i, O = vnf_{i+1}, A = A_i^q, C = C_i^q, D = allow \rangle$$

The previous rule states that vnf_i is allowed to perform the action A_i^q (implemented by the content of the traffic $\mathcal{T}_i$) over vnf_{i+1} if the context C_i^q is satisfied in the considered system. Finally, The previous rule is transformed to a DTE specification as described in Definition 3.3.

3.3.8 DTE policy enforcement

The DTE policy obtained from the transformation of the access control policy to be enforced and the network service to be deployed is enforced using a special VNF we called VNF_Filter. VNF_Filter will basically analyze the traffic exchanged between the different VNFs that compose the deployed network service to evaluate the authorization of each access query. In order to allow this, we should modify (as described in Definition 3.13) the forwarding graphs used to orchestrate and manage traffic through the VNFs that compose the deployed network service to make sure that these traffics pass certainly through the VNF_Filter.

Definition 3.13 (Forwarding graph modification) *Given a network service $\mathcal{S}$ composed of a set of forwarding graphs $\{fg_1, \cdots, fg_m\}$. Each forwarding graph fg_i is composed of a set of forwarding paths $\{fp_1, \cdots, fp_d\}$, and each fp_i is represented by a sequence $sq_i = \langle vnf_1^i, vnf_2^i, \cdots, vnf_{n_i}^i \rangle$ of the VNF that represents the path that should be traversed by a traffic. Each $sq_i = \langle vnf_1^i, vnf_2^i, \cdots, vnf_{n_i}^i \rangle$ of a forwarding path fp_i will be modified as following:*

$$sq_i = \langle vnf_1^i, \textbf{\textit{vnf\_filter}}, vnf_2^i, \textbf{\textit{vnf\_filter}}, \cdots, \textbf{\textit{vnf\_filter}}, vnf_{n_i}^i \rangle$$

To illustrate, let us consider a forwarding path fp composed of a sequence of three VNFs $\langle vnf1, vnf2, vnf3 \rangle$. The modification of fp according to Definition 3.13 makes

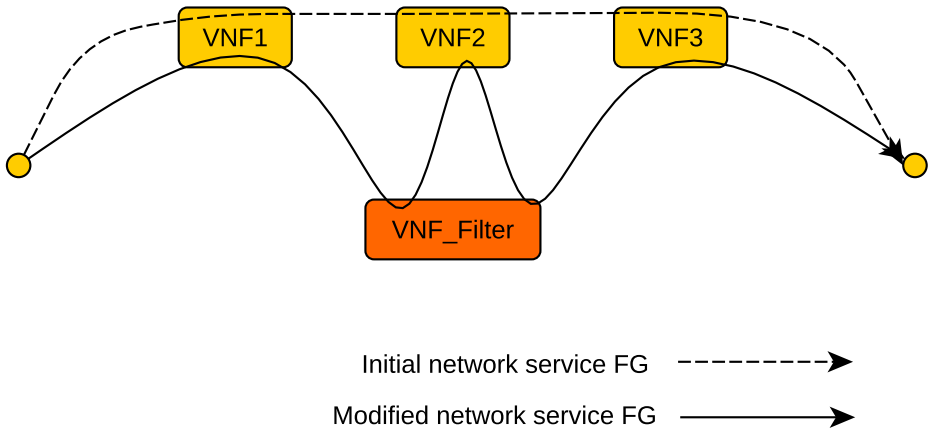


Figure 3.11 – Network Service forwarding graph modification

sure that the traffic managed by fp will pass through the VNF_Filter as shown in Figure 3.11.

The observation of all traffics exchanged between the VNFs that compose the considered network service gives VNF_Filter the ability to analyze those traffics and authorize only the ones that are allowed by the considered DTE policy.

3.4 Implementation and experimental evaluations

This section presents the implementation details of a prototype of our proposed access control model. The design architecture of the prototype implementing the proposed model is illustrated in Figure 3.12. The major functional components are described in the following.

- **OpenStack Tacker [Tac17]**: it is an official OpenStack project that orchestrates and manages infrastructure resources and maintains the lifecycle management of network services and VNF instances over the OpenStack infrastructure.
- **Access control engine (ACE)**: it is an entity that we have developed. It is responsible for parsing (1) the NFV service descriptor we are working on, (2) the security attributes, and (3) the access control policy to be eployed. Then it uses the Openstack tacker service to transform the access control policy to concrete DTE policy.

- **OpenFlow Manager of OpenDaylight (ODL)** [odl] is an open-source application development and delivery platform. OpenFlow Manager (OFM) [ofm] is an application that runs on top of ODL allowing to visualize OpenFlow topologies, program network traffic flow paths and gather network traffic stats.

- **OpenStack Infrastructure**: Openstack is the open source cloud computing platform. This Infrastructure as a Service (IaaS) is open and massively scalable. OpenStack as a virtual infrastructure manager (VIM) layer is used to give a standardized interface for managing, monitoring and assessing all resources within NFV infrastructure.

In addition, Figure 3.12 illustrates the different steps that are implemented in order to deploy an access control policy on a VNF network service. In the following, more details on each step are given:

- Onboard and deploy the network service (Steps 1 and 2): In these steps, Tacker uses the network service descriptor provided as an input to onboard and deploy the network service on the OpenStack infrastructure.

- Access control policy parsing and transformation (Steps 3 and 4): In these steps, the access policy engine parses the VNF network service descriptor, the security properties associated with the different VNFs that compose the network service (e.g., security level, trust level, etc.), and the access control policy to be deployed. Then, it transforms the access control policy to a DTE policy as described in Definition 4.3.

- DTE policy refinement (Steps 5 and 6): The ACE engine queries Tacker to get the set of resources (e.g., VMs, Connection points, networks, etc.) that are used to deploy the different VNFs that compose the deployed service. Then, it refines the DTE-policy at the resources level of the NFV service.

- Policy enforcement (Steps 7, 8, 9): To enforce the DTE policy, the ACE first uses Tacker to deploy VNF_Filter which is a special VNF that implements a DTE engine we developed in python [Smi]. Second, it loads the refined DTE policy to be enforced over the deployed NFV service on VNF_Filter. Third, ACE updates the forwarding graphs of the deployed network service (as illustrated in Figure 3.11) and uses OpenFlow Manager of ODL to make sure that all network flows

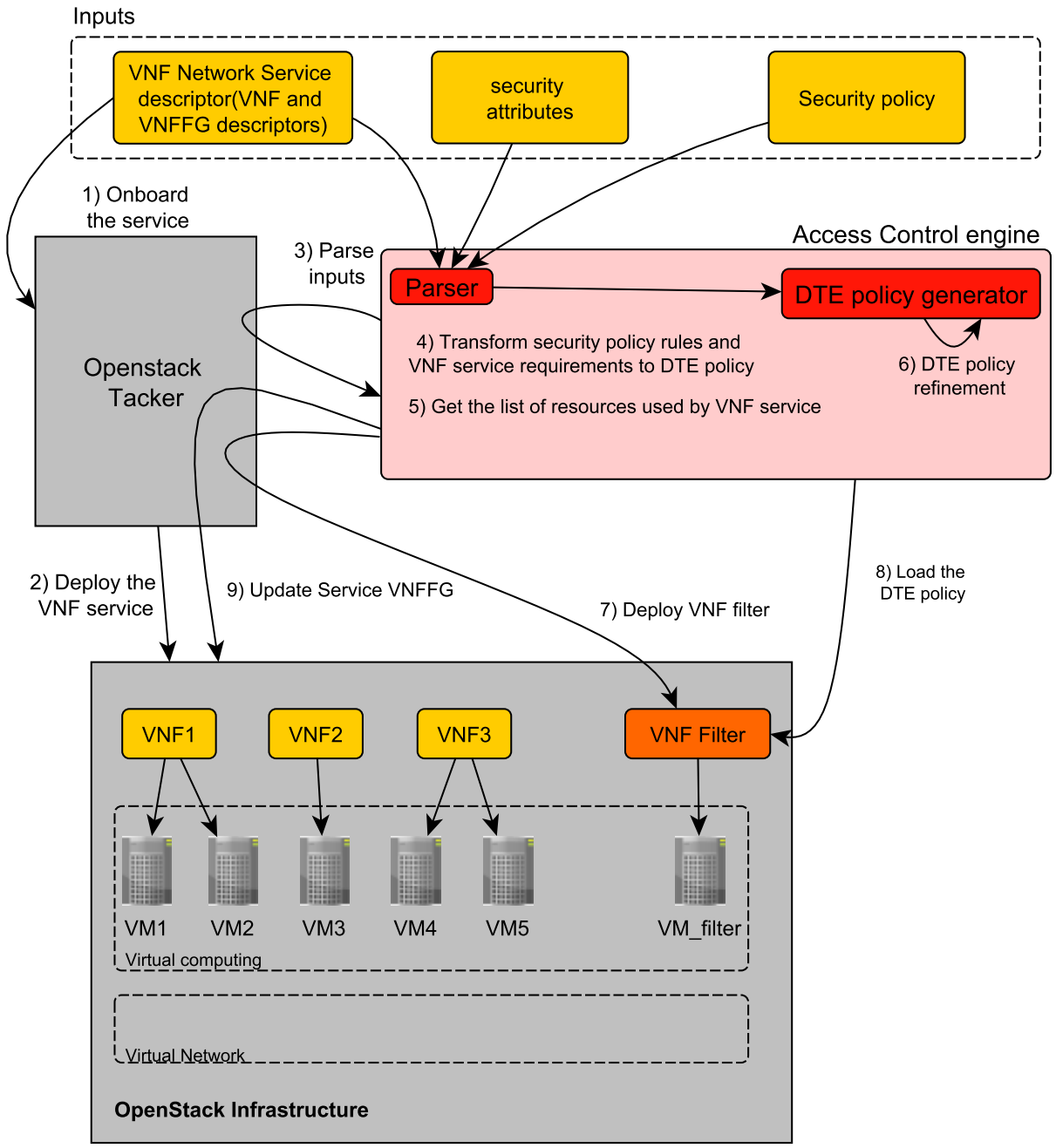


Figure 3.12 – Design architecture of the implementation of the proposed model and the operational flow of an access control policy deployment

exchanged between the VNFs that compose the deployed network service will transit through VNF_Filter. Once VNF_Filter receives a network traffic, it starts by parsing the traffic to extract its source and its destination as well as the actions that are implemented by its content. Finally, it uses the DTE engine to check whether the traffic is allowed to transit from its source to its destination i.e., the actions that are implemented by the content of the traffic are allowed to be performed by the traffic source component over the traffic destination component.

Remark 3.4.1

As we can see in the architecture that implements our approach (Figure 3.12), the deployment of the access control policy on NFV services does not require any modification of the NFV infrastructure. Indeed, it goes mainly through the *Tacker* service that is generally used to manage the NFV services. We add the *VNF-filter* which will deploy the access control policy and through the same service, we modify the forwarding graph to make sure that all the traffic goes through the added *VNF-filter* to be analyzed. Hence we positively answered to the research question RQ-5.3.

We experimentally evaluate the performance of our approach. Our access control engine prototype is hosted in a server running Linux with an Intel Xeon E5-2680 v4 Processor with 8 vCPU and 16 GB of RAM while our implementation of the VNF filter including the implementation of the DTE engine is hosted in a virtual machine running Linux having a processor with 2 vCPU and 2 GB of RAM.

In our empirical evaluation, we aim to quantify the following characteristics of our approach. First, the time needed to transform an access control policy to a DTE specification as a function of the number of rules of the considered access control policy is quantified. The obtained results are depicted in Figure 3.13.

They show that the transformation method we are proposing is quite efficient since it takes around 230 ms to transform an access control policy composed of 10^4 rules to a DTE specification. The time needed to transform a security policy to a DTE specification grows linearly in function of the number of rules in the policy. Second, we quantify the round-trip time (RTT) required for a packet as a function of the activation of our VNF_Filter (i.e., we aim to compare the RTT when our VNF_filter is used and when it is not) and the number of rules in the considered access control policy.

Figure 3.14 reports a linear growth of the measured RTT in function of the number

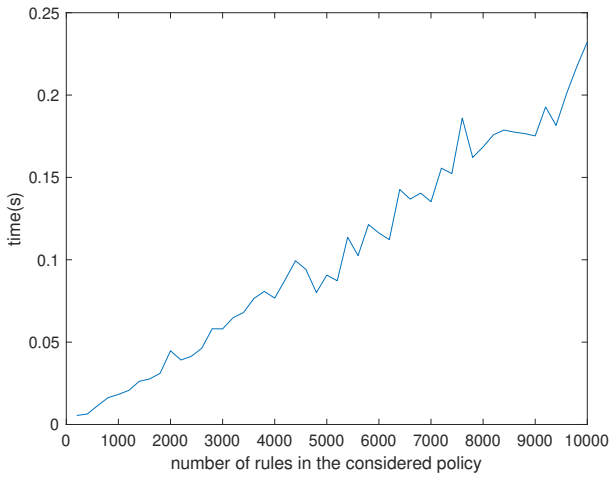


Figure 3.13 – Policy transformation time

of rules in the policy to be deployed. It shows that our implementation introduces less than 2 ms delay when a policy composed of 500 rules is considered.

3.5 Conclusion

This chapter proposes an access control as a service model to improve security management in the context of NFV services. To answer the research question RQ-5.1, we define an expressive access control policy specification model and show formally that it can be used to specify any access control requirement modeled in RBAC, ABAC, or ORBAC. Then, to answer RQ-5.2, we propose a provably correct method allowing to refine high level access control rules specified in our model to a concrete DTE specification. Finally, to answer RQ-5.3, we propose an ETSI-NFV compliant, efficient, and scalable enforcement method, as illustrated by the different conducted experimental evaluations.

Compared to existing models, our solution offers several advantages to VNF users. First, it is generic in the sense that our model allows to handle the deployment of policies expressed using most well known access control models such as RBAC, ORBAC, ABAC. Second, it complies with the ETSI-NFV infrastructure because it does not require any modification of the latter for policy deployment. The conducted experimentation shows that the implementation of the proposed model is quite efficient. The deployment of a security policy composed of 500 rules introduces less than 2 ms delay

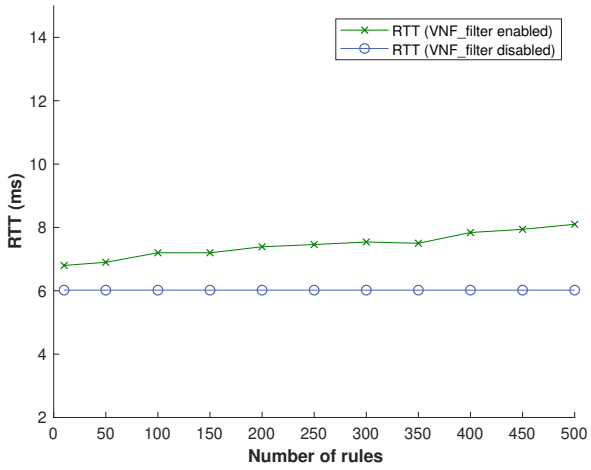


Figure 3.14 – RTT as a function of the number of rules in the access policy to be deployed

for the round-trip time of a network packet.

The solution proposed in this chapter relies on transforming the high-level access control requirements towards concrete DTE specifications. The latter are close policies where only authorizations (e.g., subject access rights, object access rights, domain transitions) are considered. As we have seen in this chapter, refining high-level closed/open policies leads to quite efficient deployable DTE specification. Nevertheless, high-level access control policies may contain both authorization and prohibition rules [ACCB07,GMW10], which may give place to conflicts and exceptions in the policy. The research questions we need to answer are then the following:

- **RQ-5.4:** Can we use the solution we proposed in this chapter to efficiently deploy high-level mixed and complex access control policies?
- **RQ-5.5:** If the previous question is answered negatively, how can we improve our solution to deal efficiently with high-level mixed and complex access control policies?

CHAPTER 4

A PRIORITY-BASED DTE FOR EXCEPTION MANAGEMENT

Contents

4.1 Introduction

As we have seen in Chapter 2, several models are proposed to manage access within and between NFV services. Unfortunately, the identified techniques suffer from at least one of the following limitations: (1) It is not clear how the transformation from the high-level policy towards a concrete deployable policy is performed (e.g., [Jae15, TCCM17]).

(2) The lack of generality by either requiring the modification of the NFV infrastructure to allow access control policy enforcement within NFV services (e.g., [PHMZ16, PTH+17]), or supporting only specific access control policy model (e.g., [PHMZ16]).

To overcome the previous limitations, in Chapter 3, we propose a Domain Type Enforcement (DTE) based formal model, allowing to handle the deployment of policies expressed using most well known access control models such as RBAC [SCFY96], ORBAC [KBB+03], and ABAC [PFMP04]. The model provides a formal and efficient method for deploying access control policies within NFV services without requiring the modification of the NFV infrastructure. However, the model proposed in the previous chapter was mainly designed to enforce closed access control policies. The latter are known to suffer from a lack of expressiveness i.e., when specific cases (e.g., exceptions) need to be excluded of general rules that should always apply [ACCB07,CCB18]. For sake of illustration, let us consider an NFV service composed of n VNFs (VNF_1, $\cdots$, VNF_n) and suppose that all VNFs except VNF_i can communicate with a specific service *s*. Using a mixed policy, the previous requirements can be expressed using the following two rules:

- VNF_i is not allowed to access to *s*.
- All VNFs are allowed to access to *s*.

However, the usage of closed policies to express the previous access control requirements will lead to a policy composed of $n-1$ high-level authorization rules, which may introduce high latency when enforced on the NFV service.

Nevertheless, despite that mixed policies are well adapted to express high-level access requirements containing exceptions, it has been shown in [ACCB07] that mixed high-level access control policies containing exceptions lead often to quite complex concrete configurations. Hence, the first research question we want to address in this chapter is as follows:

- **RQ-5.4:** Can we use the access control policy deployment solution proposed in Chapter 3 to efficiently deploy high-level mixed and complex access control policies containing exceptions?

Then, if the previous question is answered negatively, the second research question to answer is the following:

- **RQ-5.5:** How can we improve the proposed solution to deal efficiently with high-level mixed and complex access control policies containing exception?

To answer the research questions RQ-5.4 and RQ-5.5 , we investigate the management of access control exception in concrete-level DTE specification in Section 4.3. Then, in Section 4.4, we propose a solution to allow a clean and efficient deployment of complex access control policies containing exceptions and/or conflict rules on NFV services. Our model allows a clean deployment in the sense that, compared to the high level security policy[1] to be deployed, it does not introduce additional low-level rules which allows security administrator to straightforwardly understand and update the deployed concrete level policy. Our model relies on a provably correct approach for exception management in DTE specification. The conducted empirical evaluations show that the priority-based DTE model we are proposing is more efficient for enforcing big and complex policies that contain exceptions.

This chapter is organized as follows, Section 4.2 illustrates the problem addressed in this chapter with a motivating example. Section 4.3 gives some definitions for the mixed policy deployment in DTE. Section 4.4 describes the proposed priority-based DTE model. Section 4.5 defines the policy transformation towards priority-based DTE specification. Section 4.6 reports the experimental results. Finally, Section 4.7 concludes the chapter.

4.2 Motivation

Many aspects of networking are continuously growing [BBS+12] and networks are relying more and more on complex NFV services involving many VNF [LDB+21]. Typically, as the complexity of the NFV services increases, the access control policies that should be deployed on them become more and more complex. They might introduce exception rules i.e., rules that exclude some specific cases of general rules that should always apply.

Regarding the access control policy models that support mixed policy i.e., policies that contain both positive and negative authorizations, the management of exceptions can be straightforwardly accomplished through the ordering of rules or the segmentation of condition attributes [ACCB07]. Nevertheless, most promising access control

1. Security policy that is expressed using the specification model proposed in Chapter 3

models on NFV (e.g., [SECBC20, SECCB20]) are relying on the DTE model. However, this latter supports only closed policies which leads to a very complex configuration when policies containing exceptions are considered. To illustrate, let us consider the two following rules:

- r_1: Any VNF providing web server functionalities and having a low security level (e.g., a web server suffering from vulnerabilities) should be prohibited from reading and deleting records from VNF providing a highly sensitive database server.

- r_2: Any VNF providing web_server functionalities should be allowed to read, write and delete records from a VNF providing a database server.

The previous two rules ensure that any web server except those having low security level can read, write, and delete records from any database. In order to be transformed to a DTE specification, r_1 and r_2 should be rewritten into a closed policy, which give the following three rules:

- r_1^*: Any VNF providing a web server functionalities can write on VNFs providing a database server.

- r_2^*: Any VNF providing a web server functionalities except the ones that have low security level are allowed to read and delete from VNF providing a database server.

- r_3^*: Any VNF providing a web server functionalities and having low security level are allowed to read and delete records from any VNF providing a database server except the ones storing highly sensitive data.

Clearly, the previous transformation increases the number of rules and introduces new domains such as the domain containing the VNF providing a web server but does not have low security level and the domain of the VNF providing a database server but does not contain highly sensitive data. We formally show in Section 4.3.3 that, as the number of exceptions increases, the number of rules and DTE domains of the DTE specification to be deployed increases exponentially, which considerably affects the performance of the access control model.

4.3 Mixed policy deployment in DTE

In this section, we define a model for exception management in DTE-based access control policies. First, using the high-level access control policy model defined in Chapter 3, we give a formalization of exceptions in access control policies. Then, in Section 4.3.3, we propose a method for transforming arbitrary high-level access control policies containing exceptions to a DTE specification. Later, we discuss the issues related to the proposed model.

4.3.1 Mixed access control policy specification

As defined in Defintion 3.4, an access control policy is composed of a set of access control rules $\{r_1, \cdots, r_i\}$. Each rule r_i comprises a subject S_i, an action A_i, an object O_i, a context C_i and a decision D_i.

For sake of simplicity, in this chapter, we omit the usage of the context in the access control rule definition, then each rule r_i in the security policy is represented as $r_i = \langle S_i, A_i, O_i, D_i \rangle$. We stress that the previous modification will have no effect on the results that are provided in the rest of this chapter.

We extend the previous definition by adding an ordering property for the rules that compose an access control policy. That is, we suppose that the rules are evaluated from top to bottom in the order they appear in the policy. The ordering property is modeled through the association of a priority to each rule according to the order in which the rule appear in the policy. This can be formalized as follows.

Definition 4.1 (Ordered-rule policy) *Given a policy $\mathcal{P} = \{r_i | i \in [1, n]\}$ where i is the index of appearance of the rule r_i in $\mathcal{P}$. $\mathcal{P}$ is an ordered policy if for all $i, j \in [1, n]$, $i > j$ implies that the rule r_i has higher priority than r_j.*

The notion of priority associated to access control rules that compose an ordered-rule mixed policy enables to express access exception as detailed in the following section.

4.3.2 Exception in access control policy

In access control policy, exceptions are used to grant (resp. revoke) permissions (resp. prohibitions) exceptionally. For this, the *exception* rule has to be defined with a higher

priority than the *generic* rule i.e., by placing the former rule before the latter in the considered ordered-rule policy. Two types of exceptions can be distinguished:

- **Full exception**: A full exception authorization (resp. prohibition) occurs when the *exception* rule is totally included in the *generic* prohibition (resp. authorization) rule. To illustrate, let us consider the two rules r_1 and r_2 defined in Section 4.2. The rule r_1 is totally included in the rule r_2 as illustrated in Figure 4.1.

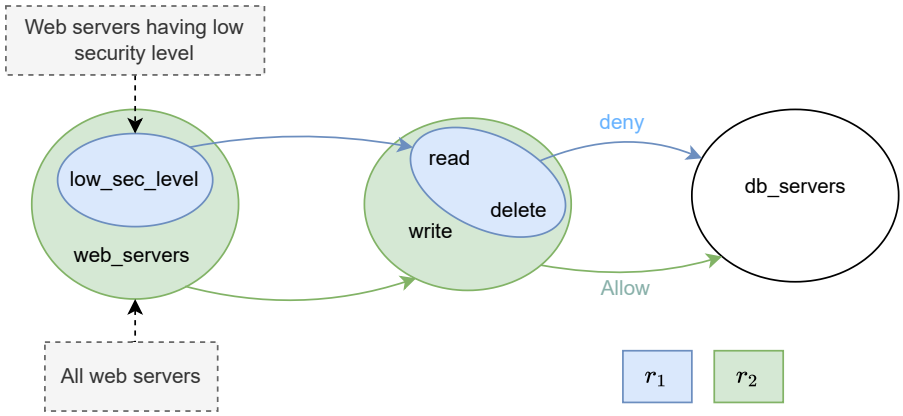


Figure 4.1 – Full exception example

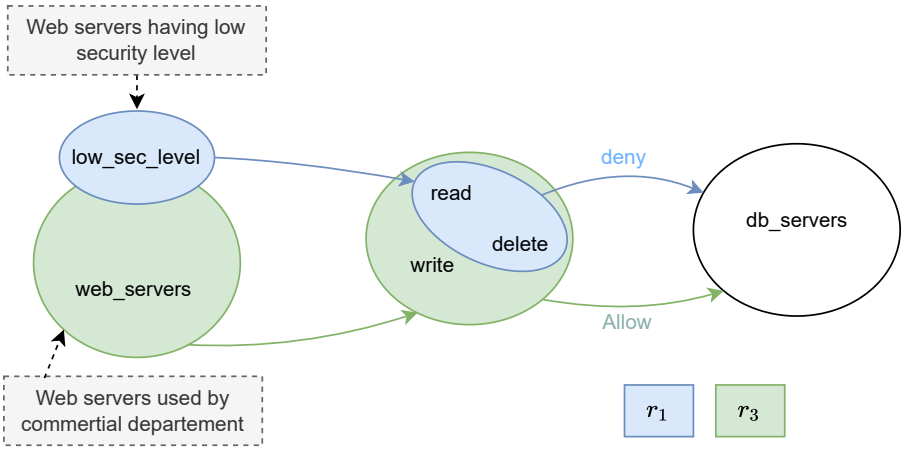


Figure 4.2 – Partial exception example

- **Partial exception**: As illustrated in Figure 4.2, a partial exception occurs when there is an intersection between a permission and a prohibition rule. As an illustration, the rule r_1 and the rule r_3 form a partial exception.

– r_3: A VNF providing web server functionalities used by the commercial department should be allowed to read, write, insert and delete records from any VNF providing a database server used by the same department.

Together, r_1 and r_3 ensure that a web server used by the commercial department except those having low security level are allowed to read, write, insert and delete records from any database server used by the same department.

Definition 4.2 (Exception) *Given an ordered-policy $\mathcal{P} = \{r_i | i \in [1, n]\}$, where i is the index of appearance of the rule r_i in the policy $\mathcal{P}$. Two rules r_i and r_j of $\mathcal{P}$ form an exception if and only if the following conditions hold:*

(i) $i < j$

(ii) $S_i \cap S_j \neq \emptyset \ \wedge \ O_i \cap O_j \neq \emptyset \ \wedge \ A_i \cap A_j \neq \emptyset \ \wedge \ D_i \neq D_j$

(iii) $S_j \nsubseteq S_i \ \vee \ A_j \nsubseteq A_i \ \vee \ O_j \nsubseteq O_i$

The exception represented by two rules r_i and r_j is denoted by $\mathcal{E}(r_i, r_j)$.

Informally, in the previous definition, condition (i) states that the appearance index of r_i should be less than the one of r_j i.e., the priority of the rule r_i is higher than the one of the rule r_j. Condition (ii) states first that the intersection between the subjects, the intersection between the objects, and the intersection between the actions of r_i and r_j are not empty. Second, the decisions of the two rules r_i and r_j are not equal i.e., the couple (r_i, r_j) is composed of an authorization rule and a prohibition rule. Finally, condition (iii) is used to make sure that the rule r_j is not shadowed. The rule r_j is shadowed by r_i if the latter matches all the access queries that match the former, such that r_j will never be activated.

We note also that, in our notation, the two arguments (binary) predicate $\mathcal{E}(\cdot, \cdot)$ is not commutative since the rule representing by the left side argument has higher priority than the rule represented by the right side argument.

Proposition 4.3.1

The definition 4.2 models both full and partial exceptions.

The proof of the previous proposition is straightforward and, hence, omitted.
In the sequel, we use the term exception for both a full or a partial exception.

4.3.3 Exception Management in DTE

As mentioned before, DTE-based systems are relying on closed policies. Hence, in order to deploy a mixed policy in a DTE-based system, we have to transform the mixed policy to be deployed to a closed policy. That is, only positive authorizations should be considered and all negative authorizations should be eliminated from the initial mixed policy. However, since there might be exceptions in the policy to be deployed, we need to make sure that these exceptions will be correctly enforced by the closed policy.

Exception transformation

In the following, we propose a method for transforming an exception represented by a negative and a positive authorizations towards a set of positive authorizations. Then, we prove its correctness.

Definition 4.3 *Given an exception* $\mathcal{E}(r_1, r_2)$ *such that* $r_1 = \langle S_1, A_1, O_1, D_1 \rangle$ *and* $r_2 = \langle S_2, A_2, O_2, D_2 \rangle$*. The exception is transformed towards a set of positive authorizations* $\mathcal{E}^*$ *as following:*

(i) if $D_1 = allow$ *and* $D_2 = deny$*,* $\mathcal{E}^* = \{r_1\}$

(ii) if $D_1 = deny$ *and* $D_2 = allow$*,* $\mathcal{E}^* = \{r_1^*, r_2^*, r_3^*\}$ *such that:*

- $r_1^* = \langle S_1^* := S_1, A_1^* := A_1, O_1^* := O_2 \backslash O_1, D_1^* := allow \rangle$
- $r_2^* = \langle S_2^* := S_2 \backslash S_1, A_2^* := A_1, O_2^* := O_2, D_2^* := allow \rangle$
- $r_3^* = \langle S_3^* := S_2, A_3^* := A_2 \backslash A_1, O_3^* := O_2, D_3^* := allow \rangle$

where $X \backslash Y$ *is used to denote the difference between the two sets* X *and* Y*.*

The case *(i)* (resp. *(ii)*) defines the transformation of exceptions in which the positive (resp. negative) authorization rule has higher priority than negative (resp. positive) authorization rule.

As we can see in Definition 4.3, the rules that compose the exception-free policy $\mathcal{E}^*$ are expressed using the '\' operator. The following proposition gives a formal definition of the aforementioned operator as a function of the properties that characterize the two sets of elements (e.g., a set of subjects or a set of objects) over which the operator is used.

Proposition 4.3.2

Given two sets of elements Θ_1 and Θ_2 characterized respectively by the two sets of properties $\mathcal{P}_1$ and $\mathcal{P}_2$, the set of elements $\Theta_* := \Theta_1 \backslash \Theta_2$ is characterized by the set of properties $\mathcal{P}^*$ such that:

$$\mathcal{P}^* = \Big\{ p \cup \ not\ p' \mid p \in \mathcal{P}_1 \cap \mathcal{P}_2 \ \text{ and } \ p' \in \mathcal{P}_2 - \mathcal{P}_1 \Big\}$$

Example 4.1 *To illustrate the exception transformation formalized in Definition 4.3, let us consider the two rules r_1 and r_3 we used respectively in Sections 4.2 and 4.3.2 :*

- *r_1: Any VNF providing web server functionalities and having a low security level (e.g., a web server suffering from vulnerabilities) should be prohibited from reading and deleting records from VNF providing a highly sensitive database server.*
- *r_3: A VNF providing web server functionalities used by the commercial department should be allowed to read, write, insert and delete records from any VNF providing a database server used by the same department.*

According to our policy specification model (Definition 3.4), the previous two rules are formalized as follows:

- $r_1 = \langle S_1 := \{func : web\_server, sec\_level : low\}, A_1 := \{read, delete\}, O_1 := \{func : database\_server, sec\_level : high\}, D_1 := deny \rangle$
- $r_3 = \langle S_3 := \{func : web\_server, used\_by : commercial\_dep\}, A_3 := \{read, write, insert, delete\}, O_3 := \{func : database\_server, used\_by : commercial\_dep\}, D_3 := allow \rangle$

According to the exception transformation method formalized in Definition 4.3, the previous two rules will be transformed to the following authorizations that are depicted in Figure 4.1.

- $r_1^* = \langle S_1^* := \{func : web\_server, sec\_level : low\}, A_1^* := \{read, delete\}, O_1^* := \{func : database\_server, used\_by : commercial\_dep, \textbf{not}\ sec\_level : high\}, D_1^* := allow \rangle$

- $r_2^* = \langle S_2^* := \{func : web\_server, used\_by : commercial\_dep, \textbf{not}\ sec\_level : low\}, A_2^* := \{read, delete\}, O_2^* := \{func : database\_server, used\_by : commercial\_dep\}, D_2^* := allow\rangle$

- $r_3^* = \langle S_3^* := \{func : web\_server, used\_by\_commercial\_dep\}, A_3^* = \{write, insert\}, O_3^* := \{func : database\_server, used\_by\_commercial\_dep\}, D_3^* := allow\rangle$

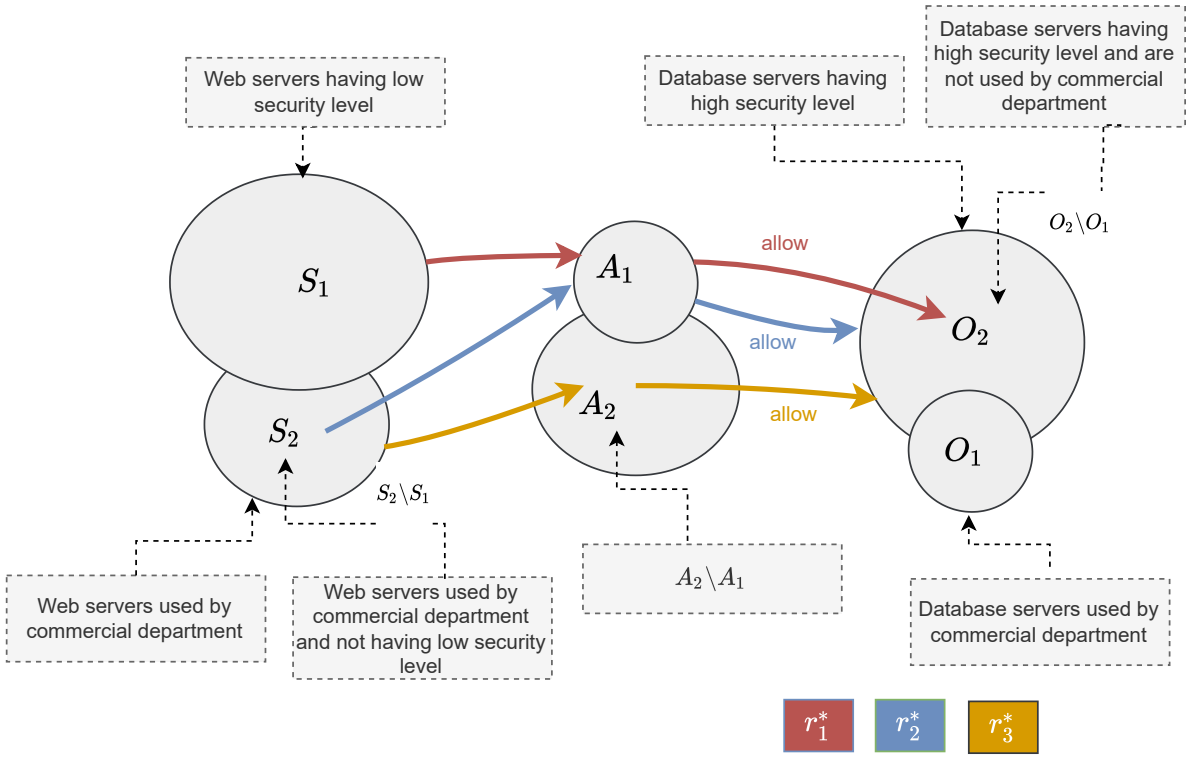


Figure 4.3 – Exception transformation example

To prove the correctness of the transformation defined in the previous definition, we first introduce the concept of access query in Definition 4.4 and define the correctness of exception transformation in Definition 4.5.

Definition 4.4 (Access Query) *An access query $\mathcal{AQ}$ is represented by the triplet $\langle S, A, O\rangle$ where S denotes the subject performing the query, A is the set of requested actions, and O denotes the object over which the query is performed.*

Definition 4.5 (Exception transformation correctness) *Given an exception $\mathcal{E}(r_1, r_2)$ and its corresponding transformation $\mathcal{E}^*$ as described in Definition 4.3. The transformation is correct if and only if, for any access query AQ, the following conditions hold:*

- *If $\mathcal{AQ}$ is allowed by $\mathcal{E}(r_1, r_2)$, then it is allowed by $\mathcal{E}^*$.*
- *If $\mathcal{AQ}$ is denied by $\mathcal{E}(r_1, r_2)$, then it is denied by $\mathcal{E}^*$.*

Theorem 4.3.1

The exception transformation method proposed in Definition 4.3 is correct.

Proof: The proof is by contradiction. We will suppose that, in both cases (i) and (ii) of Definition 4, there is an access query $AQ = \langle S^*, A^*, O^* \rangle$ that is (a) allowed by $\langle r_1, r_2 \rangle$ and denied by $\mathcal{E}^*$, and (b) denied by $\langle r_1, r_2 \rangle$ and allowed by $\mathcal{E}^*$. Then we will show that in both cases we get a contradiction.

Case 1: if $D_1 = allow$ and $D2 = deny$

(a) AQ is allowed by $\langle r1, r2 \rangle$ implies that

$$S^* \subseteq S_1 \wedge A^* \subseteq A_1 \wedge O^* \nsubseteq O_1 \tag{4.1}$$

while AQ is denied by $\mathcal{E}^*$ implies

$$S^* \nsubseteq S_1 \vee A^* \nsubseteq A_1 \vee O^* \tag{4.2}$$

Or, $S^* \subseteq S_1 \wedge A^* \subseteq A_1 \wedge O^* \nsubseteq O_1$ contradicts $S^* \nsubseteq S_1 \vee A^* \nsubseteq A_1 \vee O^* \nsubseteq O_1$ which means that Formula (4.1) contradicts Formula (4.2). Therefore the case (a) gives a contradiction.

(b) AQ is denied by $\langle r1, r2 \rangle$ implies that

$$(S^* \nsubseteq S_1 \vee A^* \nsubseteq A_1 \vee O^* \nsubseteq O_1) \wedge (S^* \subseteq S_2 \wedge A^* \subseteq A_2 \wedge O^* \subseteq O_2) \tag{4.3}$$

while AQ is allowed by $\mathcal{E}^*$ implies

$$S^* \subseteq S_1 \wedge A^* \subseteq A_1 \wedge O^* \nsubseteq O_1 \tag{4.4}$$

Formula (4.3) contradicts Formula (4.4) which prove that condition (i) of Definition 4 holds.

Case 2: if $D_1 = deny$ and $D2 = allow$

(a) AQ is allowed by $\langle r_1, r_2 \rangle$ implies that

$$(S^* \nsubseteq S_1 \vee A^* \nsubseteq A_1 \vee O^* \nsubseteq O_1) \wedge (S^* \subseteq S_2 \wedge A^* \subseteq A_2 \wedge O^* \subseteq O_2) \tag{4.5}$$

while AQ is denied by $\mathcal{E}^*$ implies

$$\begin{aligned}(S^* \nsubseteq S_1 \vee A^* \nsubseteq A_1 \vee O^* \nsubseteq O_2 \backslash O_1) \wedge (S^* \nsubseteq S_2 \backslash S_1 \vee A^* \nsubseteq A_1 \vee O^* \nsubseteq O_2) \wedge \\ (S^* \nsubseteq S_2 \vee A^* \nsubseteq A_2 \backslash A_1 \vee O^* \nsubseteq O_2)\end{aligned} \tag{4.6}$$

In Formula (4.6), any conjunctive clause including $O^* \nsubseteq O_2$, or $S^* \nsubseteq S_2$ will be in contradiction with Formula (4.5). In addition, any conjunctive clause containing $A^* \nsubseteq A_2 \backslash A_1 \wedge A^* \nsubseteq A_1$, $S^* \nsubseteq S_2 \backslash S_1 \wedge S^* \nsubseteq S_1$, or $O^* \nsubseteq O_2 \backslash O_1 \wedge O^* \nsubseteq O_1$ is in contradiction respectively with the clauses $A^* \subseteq A_2$, $S^* \subseteq S_2$, or $O^* \subseteq O_2$ of Formula (4.5). Then, it remains only to show that $A^* \nsubseteq A_2 \backslash A_1 \wedge S^* \nsubseteq S_2 \backslash S_1 \wedge O^* \nsubseteq O_2 \backslash O_1$ is in contradiction with Formula (4.5). Since, $A^* \nsubseteq A_1 \wedge A^* \subseteq A_2$ contradicts $A^* \nsubseteq A_2 \backslash A_1$, $S^* \nsubseteq S_1 \wedge S^* \subseteq S_2$ contradicts $S^* \nsubseteq S_2 \backslash S_1$, and $O^* \nsubseteq O_1 \wedge O^* \subseteq O_2$ contradicts $O^* \nsubseteq O_2 \backslash O_1$. Therefore the case (a) gives a contradiction.

(b) AQ is denied by $\langle r_1, r_2 \rangle$ implies that

$$S^* \subseteq S_1 \wedge A^* \subseteq A_1 \wedge O^* \subseteq O_1 \tag{4.7}$$

while AQ is allowed by $\mathcal{E}^*$ implies

$$\begin{aligned}(S^* \subseteq S_1 \wedge A^* \subseteq A_1 \wedge O^* \subseteq O_2 \backslash O_1) \vee (S^* \subseteq S_2 \backslash S_1 \wedge A^* \subseteq A_1 \wedge O^* \subseteq O_2) \vee \\ (S^* \subseteq S_2 \wedge A^* \subseteq A_2 \backslash A_1 \wedge O^* \subseteq O_2)\end{aligned} \tag{4.8}$$

Since $S^* \subseteq S_2 \backslash S_1$ contradicts $S^* \subseteq S_1$, $A^* \subseteq A_2 \backslash A_1$ contradicts $A^* \subseteq A_1$, and $O^* \subseteq O_2 \backslash O_1$ contradicts $O^* \subseteq O_1$, then Formula 4.7 contradicts Formula 4.8 which prove that condition (ii) of Definition 4 holds.

□

Now that we have proved the correctness of the exception transformation method we propose in Definition 4.3, we will focus on investigating its effectiveness by evaluating the complexity of the low level exception-free concrete policy. To meet our objective, in the rest of this section, we propose an algorithm that implements our exception

transformation method. Then, we give a formal quantification of the complexity of the resulting concrete level DTE specification to evaluate the effectiveness of our exception transformation method.

The following algorithm presents a method allowing to transform a mixed policy towards a closed policy. It takes as an input the initial mixed policy to be deployed and produces a set of rules representing the closed policy that can be deployed in the DTE-system.

Input: $\mathcal{P}$ /* the policy to be deployed */

1 $closed\_policy = \emptyset$
2 $exceptions =$ **get_exception(**$\mathcal{P}$**)** /* getting the exceptions in $\mathcal{P}$ */
3 **foreach** $exception \in exceptions$ **do**
4 $\quad \mathcal{E}^* =$ **transform(**$exception$**)** /* Definition 4.3 */
5 $\quad new\_exception =$ **get_exception(**$\mathcal{E}^*, P$**)**
6 $\quad exceptions = exceptions \cup\ new\_exception$
7 $\quad closed\_policy = closed\_policy \cup\ \mathcal{E}^*$
8 **end**
9 **foreach** $rule \in \mathcal{P}$ **do**
10 $\quad$ **if** $rule \notin exceptions\ and\ rule\ is\ "allow"$ **then**
11 $\quad\quad closed\_policy = closed\_policy\ \cup\ rule$
12 $\quad$ **end**
13 **end**
14 **Return** $closed\_policy$

Algorithm 2: Policy transformation algorithm

The first loop (lines 3 to 8) transforms all the exceptions in the initial policy as described in Definition 4.3. The function `get_exception` is used to identify all the exceptions in the input policy as described in Definition 4.2. Then, the `transform` function transforms each exception as described in Definition 4.3. Finally, the `get_exception` function is used to identify the exceptions that may occurs between the rules in $\mathcal{E}^*$ and the other rules in the input policy, which will be added to the set of exceptions to transform.

In the second loop (lines 9 to 13), we add to the closed policy all the positive authorization rules that have not been part of an exception.

The number of rules in the closed policy increases exponentially on the number of exceptions in the policy to be deployed as stated by the following theorem.

Theorem 4.3.2

Let us consider an ordered policy $\mathcal{P} = \{r_1, r_2, \cdots, r_n\}$. Let us denote by Ω_i the set of rules in $\mathcal{P}$ that represent an exception of r_i (i.e., $\forall r_j \in \Omega_i : \mathcal{E}(r_j, r_i)$), and by $\mathcal{P}^*$ the closed form of $\mathcal{P}$. In the worst case:

$$\|\mathcal{P}^*\| = \sum_{i=1}^{n} \Theta_i \text{ with } \Theta_i = \left\{ \begin{array}{ll} 3^{\|\Omega_i\|} & \text{if } D_i = allow \\ 0 & \text{if } D_i = deny \end{array} \right\}$$

where $\|x\|$ and D_i denote respectively the cardinality of x and the decision of r_i.

Proof: The proof is by induction. Let us denote by $\mathcal{P}_i^*$ the closed form of $\mathcal{P}_i = \{r_1, \cdots, r_i\}$, the policy composed of the i-st rules of $\mathcal{P}$.

$n = 1$: First, let us prove that the theorem holds for $n = 1$. In this case, the policy $\mathcal{P} = \{r_1\}$.

- D_1 = deny: In this case $\mathcal{P}^* = \emptyset$ since the rule r_1 will be removed when $\mathcal{P}$ is transformed to a closed policy. According to the theorem, $\|\mathcal{P}^*\| = 0$ which is correct.
- D_1 = allow: In this case the closed form $\mathcal{P}^* = \{r_1\}$. In fact, since the rule r_1 does not have any exception, then $\Omega_1 = \emptyset$. Therefore, $\|\mathcal{P}^*\| = 3^{\|\Omega_1\|} = 3^0 = 1$, which is correct.

$n = k + 1$: Let us now suppose that the theorem holds for $n = k$ and prove that it holds for $n = k + 1$. Two cases are to be considered:

- D_{k+1} = deny: In this case, the rule r_{k+1} will not change anything on the closed form of the policy $\mathcal{P}$. This is mainly due to the fact that the transformation of an exception of the form $\mathcal{E}(*, r_{k+1})$ does not introduce any new rule to the policy (see case (i) of Definition 4). Then, Theorem 2 is correct for this case since, according to the latter $\|\mathcal{P}^*\| = \|\mathcal{P}_k^*\| + 0 = \|\mathcal{P}_k^*\|$.
- D_{k+1} = allow: Since r_{k+1} has $\|\Omega_{k+1}\|$ exceptions. Let us suppose that $\Omega_{k+1} = \{\mathcal{E}_1^{k+1}, \cdots, \mathcal{E}_m^{k+1}\}$. According to Definition 4, in the worst case, $\mathcal{E}_m^{k+1}$ introduces

three new rules to the policy. And, in the worst case, each of these three added rules will form a new exception with the denial rules of $\mathcal{E}_1^{k+1} \cdots, \mathcal{E}_{m-1}^{k+1}$. Then, in the worst case, we end up with $3^{\|\Omega_{k+1}\|}$ new rules when r_{k+1} is added to $\mathcal{P}_k$. As a result, $\|\mathcal{P}^*\| = \|\mathcal{P}_k^*\| + 3^{\|\Omega_{k+1}\|} = \sum_{i=1}^{k+1} 3^{\|\Omega_i\|}$, which conclude the proof.

□

Remark 4.3.1

The approach we describe in this section allows to transform an arbitrary mixed policy towards a closed policy that can be straightforwardly deployed in a DTE-based access control system by considering each subject (resp. object) of a rule as DTE domain (resp. DTE type). Hence, as we show in Theorem 4.3.2 that the number of rules in the exception-free closed policy grows exponentially in the number of exceptions, then the number of DTE types and domains grow also exponentially in the number of exceptions in the policy to be deployed. This makes it difficult for security administrators to read, understand, update and maintain the deployed DTE specifications, which answers negatively the research question RQ-5.4. The previous results are experimentally validated by the conducted evaluations (See Section 4.6).

4.4 Priority-based DTE

To overcome the complexity problem shown in the previous section, we extend our policy enforcement model by proposing a priority-based DTE model that we describe in this section.

In our priority-based DTE model, we extend the concept of DTE domain transition by adding two key elements: a transition condition C and a priority P. The classic DTE transition and its extension are defined in the following definitions.

Definition 4.6 (DTE Transition) *A DTE transition T is represented using the triplet $\langle S_d, E, D_d \rangle$ where S_d and D_d represent respectively the source and destination domains of the DTE transition, and E represents one or many entry points that can be used to transit an access query from S_d to D_d.*

Definition 4.7 (Transition Condition) *A domain transition condition C is defined as an association between a set of actions A and a DTE type O_t. C is denoted in our model as $A_c \rightarrow O_t$. We say that C is satisfied by an access query $AQ = \langle S, A, O \rangle$ if and only if the following conditions hold:*

- $A \subseteq A_c$
- *The DTE type O_t is associated to O*

Definition 4.8 (Extended DTE Transition) *The extended DTE transition is represented by the quintuplets $\langle S_d, C, E, P, D_d \rangle$ where S_d, E and D_d are the same elements used in the Definition 4.6 and C and P are respectively the transition condition and priority of the extended transition applicability.*

Semantically, the extended DTE transition states that if an access query AQ is created in the source domain S_d (i.e., the subject performing the access belongs to S_d) and satisfies the condition C, then AQ can use any entrypoint in E to transit from S_d to the destination domain D_d. We formalize the previous statements as follows.

Remark 4.4.1

In our model, we suppose that a different priority value is associated to each transition, i.e., no more than one transition can have the same priority.

Definition 4.9 (Possible Transition) *Given a transition $T = \langle S_d, C, E, P, D_d \rangle$. T is a possible transition for an access query $AQ = \langle S, A, O \rangle$ if and only if the following conditions hold:*

- *The subject S of AQ belongs to S_d or AQ has transited to S_d*
- *AQ satisfies C (Defintion 4.7)*

Example 4.2 *To illustrate the concept of possible transition, let's assume that we have two access queries AQ_1 and AQ_2 such that:*

- *$AQ_1 = \langle web\_server\_1, read, db\_server\_1 \rangle$ in which the web server 1 wants to read from the database server 1*
- *$AQ_2 = \langle web\_server\_2\_high, read, db\_server\_2 \rangle$ in which the web server 2 having high security level wants to read from the database server 2.*

Let us suppose that the considered DTE specification contains:

- web_server_d*: a domain containing VNFs providing a web server functionalities,*
- $web\_server\_h\_d$*: a domain containing VNFs providing a web server functionalities and having high security level,*
- db_server_t*: a type associated to VNFs providing database functionalities*

Hence, as illustrated in Figure 4.4, web_server_1 *belongs to* web_server_d*,* $web\_server\_2\_high$ *belongs to both* web_server_d *and* $web\_server\_h\_d$*, and* db_server *belongs to* db_server_d *and is associated to the DTE type* db_server_t*.*

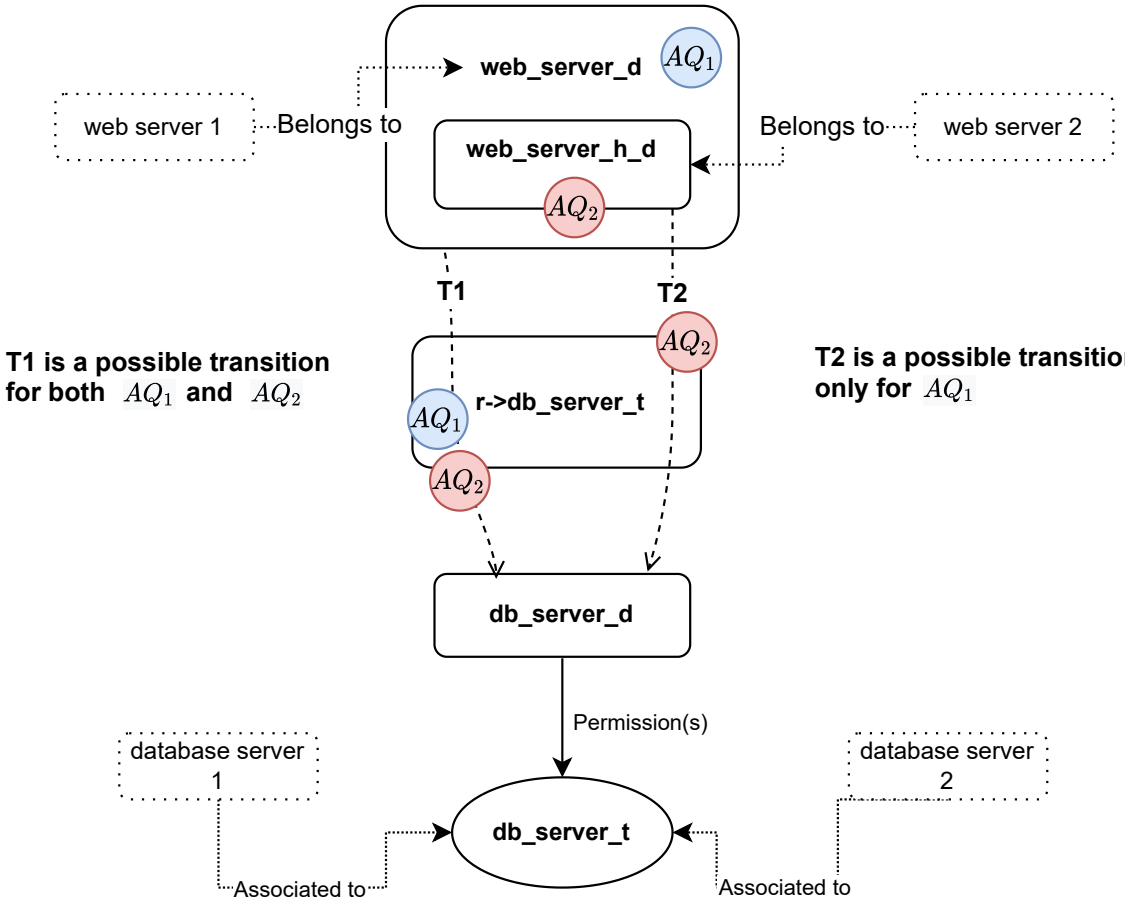


Figure 4.4 – Possible DTE transition for considered access queries in Example 4.2

Let us now suppose that we have the following two extended DTE transitions:

- $T_1 = \langle web\_server\_d, db\_server\_t : read, /bin/db\_reader, p_1, db\_server\_d \rangle$ *that says that the transition of an access query from the domain* web_server_d *to the domain* db_server_d *is possible if: (1) the access query is running in* web_server_d*, (2) the access query aims to read from a resource in* db_server_t*, and (3) the access query executes the entrypoint* $"/bin/db\_reader"$*.*

- $T_2 = \langle web\_server\_h\_d, db\_server\_t : read, /bin/db\_reader, p_2, db\_server\_d \rangle$ *that says that the transition of the access query from the domain* $web\_server\_h\_d$ *to the domain* db_server_d *is possible if: (1) the access query is running in the domain* $web\_server\_h\_d$*, (2) the access query aims to read from a resource in* db_server_t*, and (3) the access query executes the entrypoint* $"/bin/db\_reader"$*.*

According to the Definition 4.9, the two transitions T_1 *and* T_2 *are possible for the access query* AQ_1 *since* $web\_server1$ *belongs to both domains* web_server_d *and* $web\_server\_h\_d$ *and* AQ_1 *satisfies all the three requirements that are defined in Definition 4.9 while only the transition* T_1 *is possible for the access query* AQ_2*.*

As we can remark in the previous example, an access query can have several possible transitions. Hence, to help the system choosing the right (according to a specific property) transition to choose, we use the priority attribute P that we introduced in the extended DTE transition definition (Definition 4.8) as follows.

Definition 4.10 (Prioritized Transition) *Given an access query* AQ *and a set of* AQ*'s possible transitions* $T_1, \cdots, T_n$*. The transition* T_j $(j \in [1, n])$ *is the prioritized transition for* AQ *if and only if:*

$$\forall (i \in [1, n]) \text{ and } i \neq j : P_j \geq P_i$$

where P_i *is the priority associated to* T_i*.*

Informally, the previous definition states that the prioritized transition for AQ is the possible transition that has the highest priority.

At this level, we have all the ingredients to define how mixed and high level access control policies containing exception(s) can be transformed to a concrete and effective (i.e., low complexity specification and enforcement) priority-based DTE specification.

4.5 A new policy enforcement model

In this section, we first propose a method for transforming a mixed access control policy towards a priority-based DTE specification. Then, we show how access queries are evaluated by the concrete level priority-based DTE specification. Afterwards, we prove the correctness of transformation and enforcement methods.

4.5.1 Policy transformation towards priority-based DTE

Given a high-level mixed policy $\mathcal{P} = \{r_1, \cdots, r_n\}$. $\mathcal{P}$ is transformed to a priority-based DTE specification by performing the following steps, for each rule $r_i = \langle S_i, A_i, O_i, D_i \rangle \in \mathcal{P}$, where i is the index of r_i in $\mathcal{P}$.

- **step 1:** Define a domain S_i_d which will contain all entities in the system that have the same properties used to characterize the subject S_i.

- **step 2:** Define a domain O_i_d which will contain all entities in the system that have the same properties used to characterize the object O_i.

- **step 3:** Define a type O_i_t which will be associated with objects that share the same properties with the access query.

- **step 4:** Define a transition T_i that transits access queries from S_i_d to O_i_d. Then, define the transition condition C_i associated to T_i as $C_i : A_i \rightarrow O_i\_t$.

- **step 5:** If r_i is an authorization query (i.e., the decision $D_i = allow$), then give to access queries that are in O_i_d and coming from S_i_d the possibility to execute the actions A_i in O_i_t. If r_i is a prohibition query (i.e., the decision $D_i = denied$), then, there is nothing to do in this step.

- **step 6:** Set the priority P_i of the transition T_i to $n - i$.

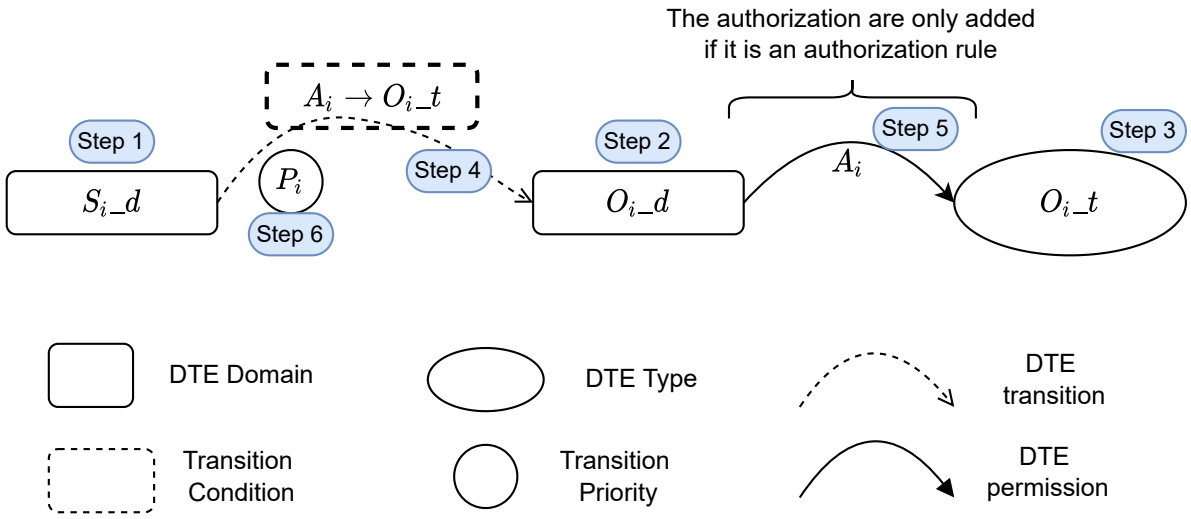


Figure 4.5 – Rule transformation.

Example 4.3 *To illustrate the previous transformation, let's take the following access control policy P that is composed of the following rules:*

- $r_1 = \langle mail\_server, \{read, write\}, ftp\_server, allow\rangle$ *which authorizes a mail server to read and write information from an ftp server.*
- $r_2 = \langle web\_server\_low, \{write\}, ftp\_server, deny\rangle$ *which denies any web server having low security level to write information on ftp server.*
- $r_3 = \langle web\_server\_low, \{read\}, db\_server, deny\rangle$ *which denies any web server having low security level to read from a database server.*
- $r_4 = \langle web\_server, \{read, write\}, db\_server, allow\rangle$ *which allows any web server to read and write on any database server.*

The transformation of the policy P towards a priority-based DTE specification is illustrated in Figure 4.6. The parts colored in green, yellow, blue and black represent the transformation of the rule r_1, r_2, r_3 and r_4 respectively.

Rule r_1: *To transform the rule r_1, first, we create the DTE domain `mail_server_d` which will contain all VNF providing mail server functionalities. Then, we create a second domain `ftp_server_d` which will contain all VNF providing ftp server functionalities. Afterwards, we create the type `ftp_server_t` which will be associated to all VNF*

providing ftp server functionalities. Then, we create the domain transition T_1 allowing to transit access queries from `mail_server_d` to `ftp_server_d` and define the transition condition C_1 to be $rw \rightarrow ftp\_server\_t$. Next, we define a DTE authorization allowing any access query in `ftp_server_d` that transits from `mail_server_d` to perform the `read` and `write` operation overs any object associated to `ftp_server_t`. Finally, we assign to 3(=4-1) the priority P_1 of T_1.

Rule r_2 : *To transform the rule r_2, first, we create the DTE domain `web_server_low_d` which will contain all VNF providing web server functionalities but having a low security level. Then, as r_1 and r_2 are defined over the same object `ftp_server`, we use the domain `ftp_server_d` and the type `ftp_server_t` that have been created during the transformation of r_1. Then, we create the domain transition T_2 allowing to transit access queries from `web_server_low_d` to `ftp_server_d` and define the transition condition C_2 to be $w \rightarrow ftp\_server\_t$. However, r_2 is a prohibition query, so we do not give any permission for access queries transiting from `web_server_low_d` to `ftp_server_d` on the objects associated with `ftp_server_t`. Finally, we assign to 2(=4-2) the priority P_2 of T_2.*

Rule r_3 : *The rule r_3 is to be transformed similarly to r_2. First, since r_2 and r_3 are defined over the same subject `web_server_low`, then we use the same DTE domain `web_server_low_d` defined during the transformation of r_2. Then, we create the DTE domain `db_server_d` which will contain all VNF providing database server functionalities. Afterwards, we create the type `db_server_t` which will be associated to all VNF providing ftp server functionalities. Then, we create the domain transition T_3 allowing to transit access queries from `web_server_low_d` to `db_server_d` and define the transition condition C_3 to be $r \rightarrow db\_server\_t$. Then, similarly to r_2, since r_3 is a prohibition query, no permission is given for access queries transiting from `web_server_low_d` to `db_server_d` on the objects associated with `db_server_t`. Finally, we assign to 1(=4-3) the priority P_3 of T_3.*

Rule r_4 : *The rule r_4 is to be transformed similarly to r_1. First, we create the DTE domain `web_server_d` which will contain all VNF providing web server functionalities. Then, as r_4 and r_3 are defined over the same object `db_server`, we use the domain `db_server_d` and the type `db_server_t` that have been created during the*

transformation of r_3. Then, we create the domain transition T_4 allowing to transit access queries from `web_server_d` to `db_server_d` and define the transition condition C_4 to be $rw \rightarrow db\_server\_t$. Next, we define a DTE authorization allowing any access query in `db_server_d` that transits from `web_server_d` to perform the `read` and `write` operation overs any object associated to `db_server_t`. Finally, we assign to 0(=4-4) the priority P_4 of T_4.

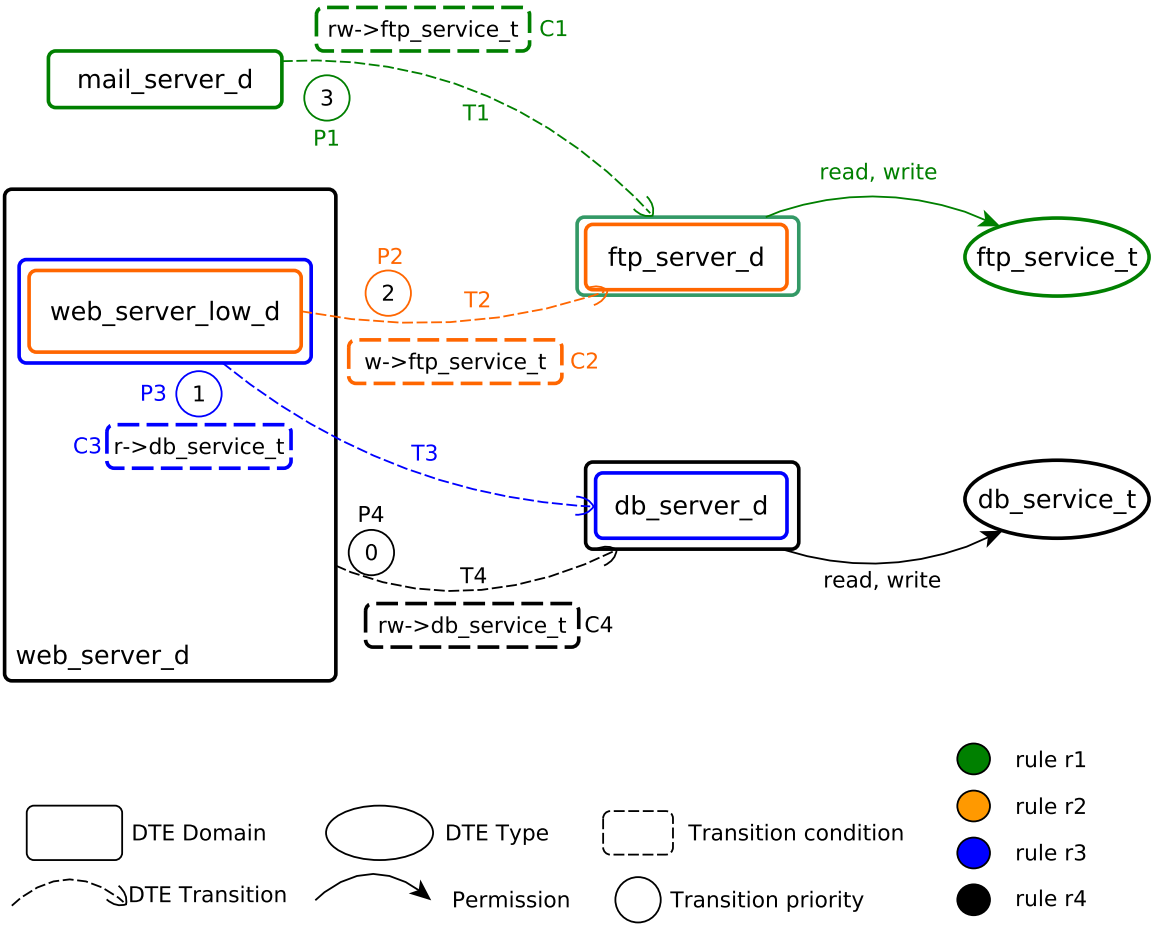


Figure 4.6 – Graphical representation of the DTE policy described in Example 4.3

Once we have defined how high-level mixed policies are transformed to priority-based DTE specification, in the next section, we show how access queries are evaluated when the concrete priority-based DTE specification is enforced.

4.5.2 Access Query Evaluation

In this section, we focus on defining how access query evaluation is performed when a priority-based DTE specification is enforced. To meet the previous objective, we first define the strategy we are considering for transiting access query between different DTE domains.

Definition 4.11 (Transition Rule) *Given an access query $AQ = \langle S, A, O \rangle$ that belongs to the DTE domain D and a set of possible DTE transitions $T_1, \cdots, T_n$. If the access query is authorized in D i.e., the entities in D are not allowed to perform the actions A on O, then AQ performs the prioritized transition (Definition 4.10).*

Informally, the previous definition states that if an access query cannot be authorized in its current DTE domain, then, it performs the DTE transition with the highest priority.

The pseudo-code in Algorithm 3 presents the procedure we use for access query evaluation when a priority-based DTE specification is enforced in the target system.

```
   Input: P^p_dte   /* the enforced priority-based DTE specification*/
          AQ = ⟨S, A, O⟩   /* the access query to be evaluated */
1  current_domain = get_domain(AQ, P^p_dte)   /* getting the DTE domain of AQ */
2  while is_authorized(AQ, current_domain, P^p_dte) == false do
3      prioritized_transition =
         get_prioritized_transition(AQ,current_domain, P^p_dte)
4      if prioritized_transition == null then
5          return fasle   /* access query is not authorized */
6      end
7      current_domain = perform_transition(AQ,prioritized_transition)
8  end
9  Return true
```

Algorithm 3: Access query evaluation under a priority-based DTE policy enforcement.

In Algorithm 3, using the function `get_domain` (line 1), we start by getting the DTE domain associated with the access query. Note that, an access query belongs initially to the DTE domain associated with the subject performing the access query. Then, as long as the current domain of the access query does not allow the required action A on the object O (i.e., the access query is not authorized), we get the prioritized transition

(line 3) for the access query and perform it (line 7). If no possible transition is available for the access query (line 4), then the latter will be prohibited. Otherwise, the access query will be accepted as it was able to transit to a DTE domain on which the action(s) required by the access query are allowed to be performed over the target object.

Example 4.4 *To illustrate the access query evaluation when a priority-based DTE specification is enforced, let us consider the same priority-based DTE policy used in Example 4.3, and let us suppose that we want to evaluate the following access queries:*

- $AQ_1 = \langle mail\_server1, read, ftp\_server1 \rangle$ *in which the mail server 1 wants to read from the ftp server 1.*
- $AQ_2 = \langle web\_server2\_low, write, ftp\_server1 \rangle$ *in which the web server 2 having low security level wants to write on the ftp server 1.*
- $AQ_3 = \langle web\_server2\_low, read, db\_server1 \rangle$ *in which the web server 2 having low security level wants to read from the database server 1.*
- $AQ_4 = \langle web\_server2\_low, write, db\_server1 \rangle$ *in which the web server 2 having low security level wants to write on the database server 1.*

When evaluating the access query AQ_1, the process executing AQ_1 will be initially in the domain $mail\_server\_d$ since the subject of AQ_1 ($mail\_server1$) belongs to the domain $mail\_server\_d$. The latter has only one possible transition towards ftp_server_d which is allowed to read from objects in ftp_server_t including ftp_server_1. As a result AQ_1 will be authorized.

When evaluating the access query AQ_2, the process executing AQ_2 will be initially in the domain $web\_server\_low\_d$ since the subject of AQ_2 (web_server2_low) belongs to the domain web_server_low_d and it belongs also to the domain web_server_d. Based on the Figure 4.6, the domain $web\_server\_low\_d$ have two possible transitions. However, only one transition is possible for AQ_2 since only the condition "$w- > ftp\_service\_t$" is satisfied. So following this transition, the process will transit to the domain $ftp\_service\_d$ and we know that all the access queries coming from the domain $web\_server\_low\_d$ do not have any possible action on the type $ftp\_service\_t$. As a result AQ_2 is denied.

To evaluate AQ_3, since the process is initially in the domain $web\_server\_low\_d$ and it belongs also to the domain web_server_d, so it will have two possible transitions (From

$web\_server\_low\_d$ and web_server_d), so the two conditions "$r- > db\_service\_t$" and "$rw- > db\_service\_t$" are satisfied. Here we will apply the concept of priority defined in Definition 4.10. As result, the prioritized transition is colored in blue in which the process will transit from the domain $web\_server\_low\_d$ to the domain db_server_d. However, the access queries coming from $web\_server\_low\_d$ do not have any possible action on the type $db\_service\_t$. As result AQ_3 is denied

When evaluating the access query AQ_4, the process executing AQ_4 belongs to both domains $web\_server\_low\_d$ and web_server_d. On the other hand, it will have only one possible transition because only the condition "$rw- > db\_service\_t$" is satisfied. So it will transit to the domain db_server_d. However, the access queries coming from web_server_d have the right to read and write on the type $db\_service\_t$. As a result AQ_4 is authorized.

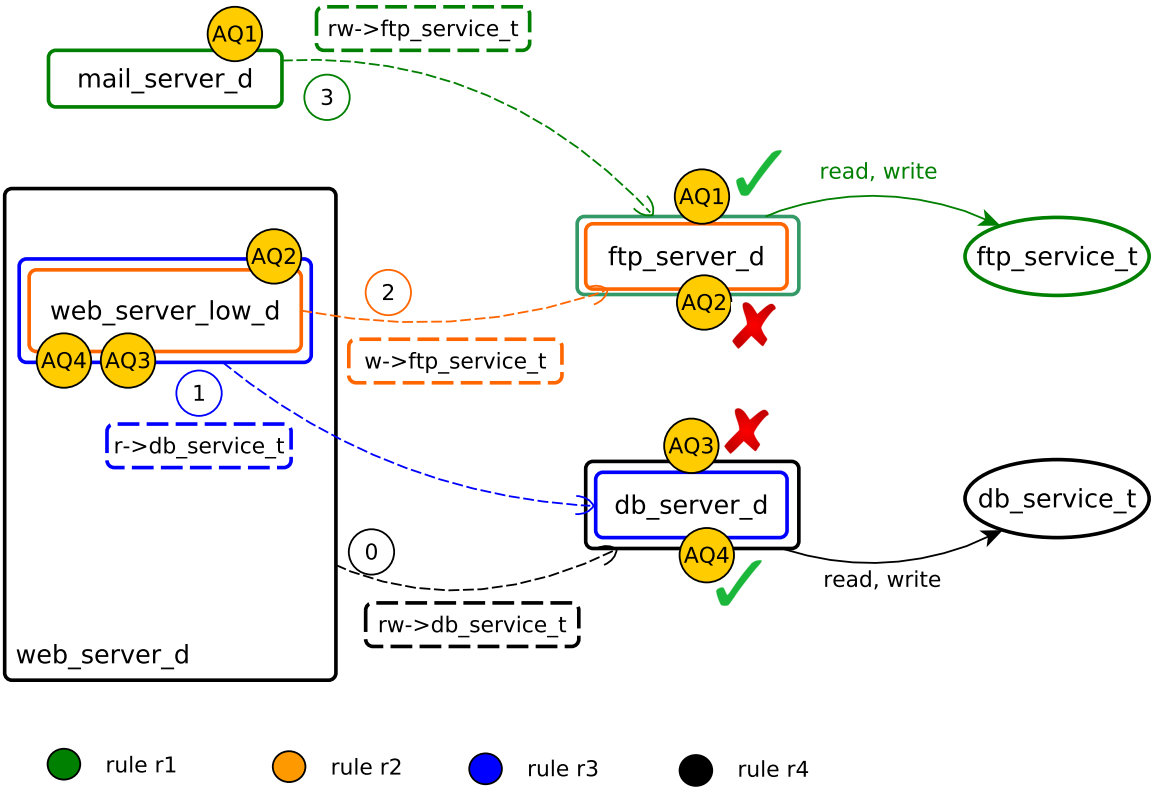


Figure 4.7 – Access Queries evaluation

Remark 4.5.1

In contrast to the exception management approach defined in Section 4.3.3, we can remark that the transformation method defined in this section does not increase the number of authorization rules in the policy to be enforced, which allows to have a constant number of DTE domains and types regardless how many exceptions are in the high-level policy to be deployed. Moreover, the complexity of the evaluation of access queries depends only on the number of rules in the high-level policies and is constant on the number of exceptions.

4.5.3 Correctness

In this section, we focus in showing the correctness of the policy transformation and query evaluation methods we propose in Sections 4.5.1 and 4.5.2 respectively.

Informally, we say that our policy enforcement (i.e., the transformation and query evaluation methods) is correct if any access query authorized (resp. prohibited) by the high-level access control policy is also authorized (resp. prohibited) by the equivalent priority-based DTE specification.

Definition 4.12 (Policy Enforcement Correctness) *Given a high-level access control policy $\mathcal{P}$ and its corresponding priority-based DTE specification $\mathcal{P}^{p}_{dte}$. Our enforcement method is correct if and only if for any access query AQ: if AQ is allowed (resp. denied) by $\mathcal{P}$, then it is allowed (resp. denied) by $\mathcal{P}^{p}_{dte}$.*

Theorem 4.5.1

The access control policy enforcement method proposed in Sections 4.5.1 and 4.5.2 is correct.

Proof: Proof is by contradiction. Let us denote by $AQ = \langle S, A, O \rangle$, $\mathcal{P}$, $\mathcal{P}^{p}_{dte}$, and $\mathbb{E}_{\mathcal{P}}$ respectively the access query to be evaluated, the high-level access control policy to be enforced, its priority-based DTE specification, and the set of exceptions in $\mathcal{P}$. According to Definition 4.12, two cases should be considered:

Case 1: AQ is authorized by $\mathcal{P}$ but not authorized by $\mathcal{P}^p_{dte}$.

Let us first focus on the evaluation of AQ by $\mathcal{P}^d_{dte}$. As we suppose that AQ is prohibited by $\mathcal{P}^d_{dte}$, this means that there is no DTE domain in $\mathcal{P}^d_{dte}$ to which AQ belongs or can transit that allows performing the action(s) A on O.

Now, let us suppose that there exists a set of $m(m \leq n)$ rules $\mathcal{R} \subseteq \mathcal{P}$ such that each rule $r \in \mathcal{R}$ matches AQ:

$$\forall r_i \in \mathcal{R} : S \in S_i \wedge O \in O_i \wedge A \in A_i$$

Hence, AQ is authorized by $\mathcal{P}$ means that the highest priority rule r^* (i.e., the rule having the lowest index in the ordered-rule policy $\mathcal{P}$) in $\mathcal{R}$ is an authorization rule.

Then, according to our policy transformation method defined in Section 4.5.1, the transformation of each $r_i = \langle S_i, A_i, O_i, D_i \rangle \in \mathcal{R}$ (depicted in Figure 4.5) creates a DTE transition allowing to transit access queries from the domain of the subject to the domain of the object considered by the rule. Then associate a priority $P_i = n - i$, where i is the index of r_i in $\mathcal{P}$.

Now, as we have supposed that there exists an authorization rule $r^* \in \mathcal{R}$ such that AQ matches r^* and that r^* has the highest priority in $\mathcal{R}$. Then, if we denote by P^* the priority associated to the DTE transition created during r^* transformation, we have:

$$\forall r_i \in \mathcal{R} \backslash \{r^*\} : P^* > P_i \tag{4.9}$$

Hence, as we suppose that AQ matches all the rules in $\mathcal{R}$, then AQ initially belongs to

$$\bigcap_{r_i \in \mathcal{R}} S_i\_d$$

the intersection of the DTE domains associated to all the subjects of the rules in $\mathcal{R}$. In addition, AQ will have m possible DTE transitions $\{T_1, \cdots, T_m\}$.

Now according to our query evaluation method, AQ will transit following the DTE transition of the rule having the highest priority r^*. Finally, as r^* is an authorization query that matches AQ, then AQ will be authorized, which contradicts our hypothesis for this case.

Case 2: AQ is not authorized by $\mathcal{P}$ but authorized by $\mathcal{P}^p_{dte}$. Similarly to the case 1, let us start by supposing that there exists a set of $m(m \leq n)$ rules $\mathcal{R} \subseteq \mathcal{P}$ such that each

rule $r \in \mathcal{R}$ matches AQ:

$$\forall r_i \in \mathcal{R} : S \in S_i \wedge O \in O_i \wedge A \in A_i$$

As we suppose that AQ is prohibited by $\mathcal{P}$, this means that the highest priority rule r^* in $\mathcal{R}$ is a prohibition rule. Then, in $\mathcal{P}^p_{DTE}$, the priority P^* associated with r^* satisfies Formula 4.10.

$$\forall r_i \in \mathcal{R} \backslash \{r^*\} : P^* > P_i$$

Similarly to the case 1, and according to the query evaluation strategy, AQ will transit following the DTE transition T^* having the highest DTE priority. However, as r^* is a prohibition rule, AQ will be prohibited, which contradicts the hypothesis considered in this case. □

4.6 Experimental Results

In this section, we experimentally evaluate the performance of our priority-based DTE model for deploying security policies on NFV services. For all conducted experiments, we use a (pseudo) randomly generated high-level policies that will be deployed on NFV services. The evaluations are conducted on a server running Linux with an Intel Xeon E5-2680 v4 Processor with 16 vCPU and 32 GB of RAM on which we install our openstack-based access control policy enforcement framework we proposed in the previous chapter (Section 3.4). We extend our framework by adding the transformation of high-level mixed access control policies towards priority-based DTE specification and the enforcement of the latter. Our objective is to compare the performance of our proposed model with the classic DTE-based access control model for NFV services proposed in Chapter 3 regarding the following characteristics:

- The number of required domains and types in the DTE specification as a function of the number of exceptions in the high-level policy to be deployed.
- The number of rules in the DTE specification as a function of the number of exceptions in the policy to be deployed.
- The time required for transforming the policy to be deployed towards a deployable DTE specification.

- The time needed to evaluate an access query as a number of exceptions in enforced policy.
- The impact on network performance.

In the sequel, we use the term classic DTE based solution to refer to the access control solution proposed in Chapter 3.

Figure 4.8 compares the growth of the number of DTE domains and types for both classic and priority-based DTE as a function of the number of exceptions in the policy to be deployed. When the latter grows by a factor of 6 (from 10 to 60), the number of DTE domains and types grow by a factor of 2 for the priority-based DTE while growing by a factor of $\approx$ 192 in the case of classic DTE. Figure 4.9 compares the number of rules required by both the classic DTE and priority-based DTE specifications as a function of the number of rules in the policy to be deployed. When the latter increases from 10 to 60, the number of rules in the priority-based DTE increases by the same factor (factor of 6) while the number of rules required by the classic DTE grows by a factor of 230.

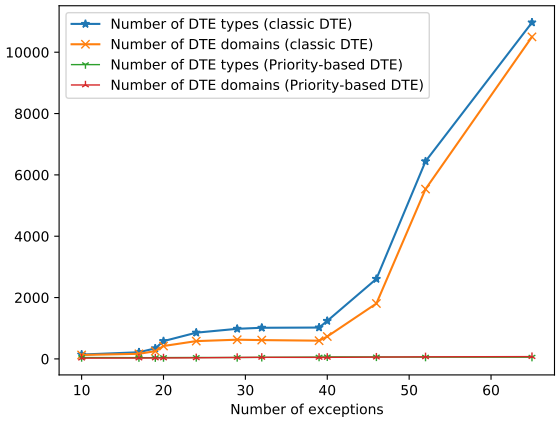


Figure 4.8 – The comparison of the growth of the number of required DTE domains and types in classic and priority-based DTE as a function of the number of exception in the policy.

Figure 4.10 compares the time required for transforming a high-level policy towards a classic DTE and priority-based DTE specifications. In the case of priority-based DTE, the time of transformation is almost constant while growing by a factor of 10^5 in the case of classic DTE as the number of exceptions increases by only a factor of 10. This is mainly due to the transformation of all exceptions of the policy (lines 3 to 8 of Algorithm

2). In fact, when an exception is transformed to a set of positive authorizations, these latter may introduce several new exceptions that need to be resolved.

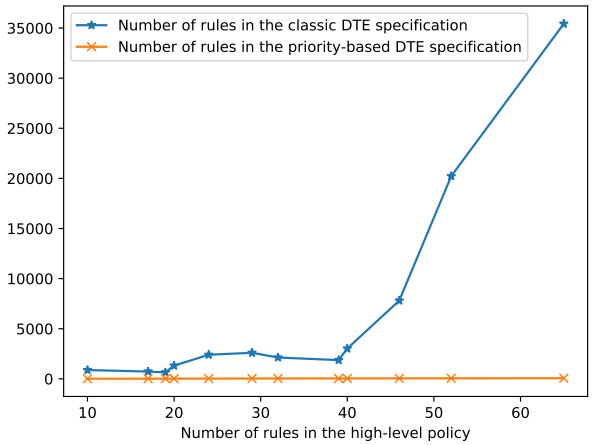


Figure 4.9 – The comparison of the number of rules in the classic DTE specification and in the priority-based DTE specification as a function of the number of rules in the policy to be deployed.

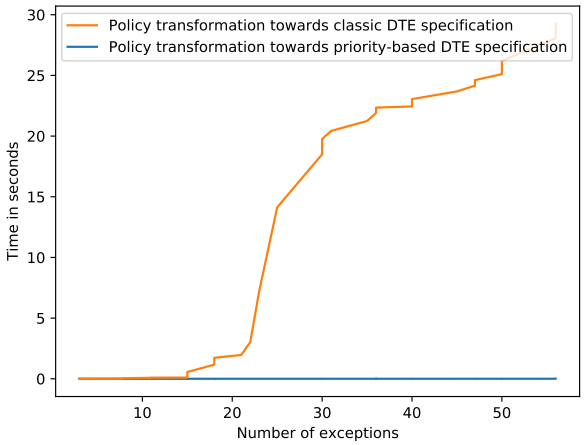


Figure 4.10 – The comparison of the time required to transform the policy towards classic DTE specification and priority-based DTE specification as a function of the number of exceptions in the high-level policy to be deployed.

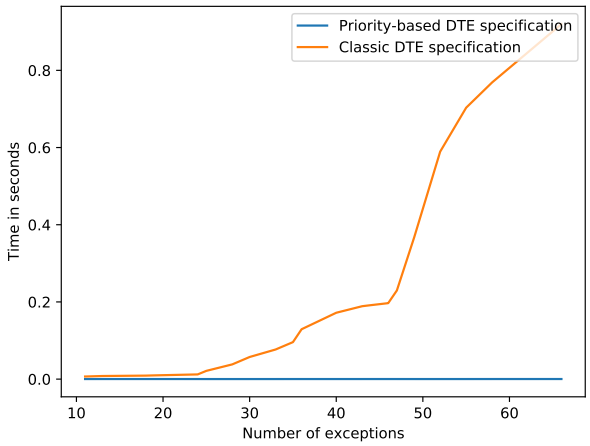
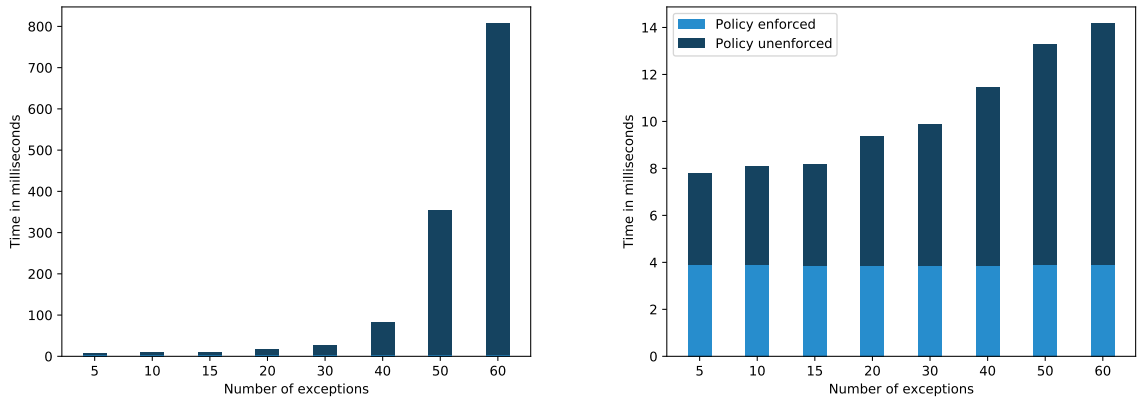


Figure 4.11 – The comparison of the access query evaluation time between the classic DTE specification and the priority-based DTE specification as a function of the number of exceptions in the policy to be deployed.

Figure 4.11 compares the time needed by both classic DTE-based model and our priority-based DTE model for the evaluation of an access query. Compared to the model proposed in Chapter 3, our priority-based DTE model drastically reduces the time needed for evaluating an access query as the number of exceptions in the policy to be deployed increases. This is due to the fact that, in the classic DTE specification, the number of domains and types grows exponentially relative to the number of exceptions. As the evaluation of an access query requires matching the domains (resp. types) with the subject (resp. object) of access query. Therefore, the time required to evaluate an access query grows exponentially according to the number of exceptions.

Finally, we compare the impact in terms of network performance on the target NFV service. Figure 4.12a (resp. 4.12b) illustrates the growth of the round-trip time (RTT) of a network request when a classic DTE (resp. our priority-based DTE) model is used as a function of the number of exceptions in the deployed policy. When the classic DTE model is used, the RTT increases by a factor of 210 (from 3,84 to 801 milliseconds) while increasing by only a factor of 4 (from 3,8 to 14.2 milliseconds) when our priority-based DTE model is used. This is mainly due to the time required to evaluate access queries with both solutions.



(a) (b)

Figure 4.12 – The comparison of the growth of the round-trip time of a network request when (a) a classic DTE policy model is used and (b) a priority-based DTE policy model is used, as a function of the number of exceptions in the enforced policy.

Remark 4.6.1

As illustrated by the conducted evaluations, the priority-based DTE model proposed in Section 4.4 of this chapter brings two main advantages compared to existing solutions. First, it does not require to create additional DTE types and domains. That is, each rule in the high-level policy will be refined using two DTE domains and a DTE type (see Section 4.5.1) which drastically reduces the complexity of reading, understanding, and updating the deployed DTE specifications. Second, the proposed solution improves drastically the efficiency of the enforcement of high-level mixed access control policies compared to existing solutions. Hence, with the previous contributions, we answer the research question RQ-5.5.

4.7 Conclusion

In this chapter, we investigate high-level mixed access control policies enforcement using DTE-based concrete level specification. In particular, we propose in Section 4.3 a first extension of the DTE-based access control enforcement solution proposed in Chapter 3 to handle high-level mixed access control policies enforcement. This first extension relies mainly on transforming high-level mixed access control policies con-

taining exceptions towards a high-level closed and exception-free access control policy. The latter is then refined to a DTE specification as described in Section 3.3.4. Then, we formally proved that the proposed solution leads to a quite complex concrete DTE specification that involves DTE domains and types whose number grows exponentially as the number of exceptions in the high-level mixed policy grows linearly.

To overcome the previous limitation, we propose a provably correct priority-based DTE access control model. We experimentally show that our proposed model is by far more efficient than the one proposed in Section 4.3 when dealing with high-level complex policies containing exceptions.

With respect to the dissertation main research question, this chapter answers RQ-5.4 and RQ-5.5 by proposing a provably correct solution for efficiently enforcing high-level mixed policy.

Access control policies enforcement on NFV service solutions we proposed in the previous chapters have considered various criteria related to access control policy specification, refinement, and deployment, such as the expressiveness of the proposed access control model, the formal modeling, efficient exception management, as well as the correctness verification of access control policy refinement and deployment. However, the proposed solutions did not take into consideration two fundamental criteria. First, the impact in terms of latency on the NFV service due to the access control enforcement. Second, the resources needed to enforce the access control policy.

In the next chapter, we focus on extending our access control enforcement over NFV service framework to optimize the impact in terms of latency that is generated by the access control policies deployment and the resources consumed by the policy enforcement points.

CHAPTER 5

OPTIMAL ACCESS CONTROL DEPLOYMENT IN NFV SERVICE

Contents

5.1 Introduction

Decoupling of network functions from proprietary appliances is a concept introduced by Network Function Virtualization (NFV). It permits network services to run on commodity cloud computing style platforms. It allows a better information sharing between their components by enabling faster deployment of new services with less risk and allowing iterative improvement of existing services. It broadens the developer ecosystem to include new entrants, while reducing network cost structure through infrastructure sharing and automation.

As we have seen in Chapter 2, security is a major concern in NFV-based networks as they are susceptible to threats that are related to network-based, virtualization, and malicious users-related vulnerabilities. Once exploited, these vulnerabilities lead often to misuse affecting the capabilities provided by NFV services as well as their end-users security.

In Chapters 3 and 4 of this dissertation, we propose an access control enforcement framework on NFV services to mitigate threats raised mainly by malicious user-related vulnerabilities (e.g., unauthorized access/privilege, unauthorized flow between VNFs). The proposed approaches have considered various criteria related to access control policy specification, refinement, and deployment, such as the expressiveness of the proposed access control model, the formal modelling and the correctness verification of access control policy refinement and deployment. However, most of the aforementioned solutions did not take into consideration two fundamental criteria. First, NFV services, as any other virtualized infrastructure, can be compromised and partially controlled[1] by an adversary. However, existing solutions as well as the ones we proposed in Chapters 3 and 4 consider only outsider adversaries that are aiming to remotely bypass the enforced policy (see Table 2.3 in Section 2.5.4). Hence, the ability to deal with both insider and outsider adversaries is an important criteria that needs to be considered when defining solutions for access control policy enforcement on NFV services. Second, the enforcement of access control policy requires resources in terms of computation and storage, and often impacts the functionality provided by the target NFV service i.e., by introducing latency due to the traffic analysis and rule enforcement. Nevertheless, all existing access control enforcement solutions on NFV infrastructure

1. If adversary totally controls the target NFV service i.e., all the VNFs that compose the NFV service are controlled by the adversary, then there is nothing to be protected by an access control approach.

including the ones we proposed in Chapters 3 and 4 does not consider minimizing both resource consumption and the impact related to access control policy enforcement (see Table 2.2 in Section 2.5.3), which we believe to be an important criteria that should be considered when designing solutions for access control policy enforcement on NFV services.

In the light of the previous observations, the research questions we aim to answer in this chapter are the following:

- **RQ-5.6:** How to improve previously proposed solutions to deal with insider adversaries? More specifically, how to correctly deploy access control policies when an unknown part of the NFV service is controlled by an adversary?

- **RQ-5.7:** How to extend our access control policy enforcement framework to minimize both the consumed resources and the impact caused by the access control policy deployment?

To answer the previous research questions, we propose the following contributions:

- We consider a strong adversary model: In contrast to our approaches proposed in Chapters 3 and 4 that only consider outsider adversary, we assume that we are dealing with an insider adversary who can control one or more nodes (VNFs) that compose the NFV service. In addition, we assume that the compromised nodes are not known.

- We propose a formal modelling of the optimal deployment of the access control policy problem allowing to model and quantify consumed resources as well as the impact in terms of latency that is to be generated by the access control policy deployment. The proposed model allows to formally prove the correctness of the access control policy deployment.

- We show that the problem of the optimal deployment of access control policies is a nonlinear multi-objective optimization problem and we use an improvement of the Non-dominated Sorting Genetic Algorithm NSGA II [WSZ+18] for solving it.

- We conduct an experiment in an emulated Internet environment, NS-3 [ns3], to evaluate the access control policy deployment solutions.

The chapter is organized as follows. Section 5.2 provides some background on multi-objective optimization and queuing theory that we use in this chapter. Section 5.3 describes the adversary model and the problem addressed in this work. Section 5.4 presents our proposed model and the formalization of the optimization problem. Section 5.5 shows how the latency is quantified. Then, in section 5.6, we study the optimization problem to identify the right resolution algorithm to use. Section 5.7 provides an overview of the implementation of our model and gives the evaluation results. Finally, Section 5.8 concludes this chapter.

5.2 Background

In this section, we present two notions that we are going to rely in the definition of the solution we propose in this chapter, namely multi-objective optimization and queuing theory.

5.2.1 Multi-objective optimization

First, we define the main notions common to any multicriteria optimization method.

Objective function: An objective function is a function that models the goal to be achieved in the optimization problem on all criteria. It is the function that must be optimized.

Parameters: A parameter of the optimization problem is a variable that expresses a quantitative or qualitative data on a dimension of the problem: cost, time, error rate, etc. These parameters correspond to the variables of the objective function. They are adjusted during the optimization process to obtain the optimal solutions. They are also called optimization variables.

Constraints: A constraint of the problem is a condition that must be respected by the decision vectors of the problem.

Solution A solution or a decision vector is a vector corresponding to the set of variables of the problem. Formally, a solution is denoted as

$$\vec{x} = \langle x_1, x_2, \cdots, x_n \rangle$$

with n is the number of variables or dimension of the problem and x_i the variable on the i^{th} dimension.

Multicriteria optimization has been widely studied in the literature [Ehr05, Gun18]. Considered optimization problems often invoke multiple performance measures or objectives, which must be optimized simultaneously. In practice, this is not always possible because the objectives may conflict, as they measure different aspects of the quality of the solution. In this case, the quality of an individual is described not by a scalar but by a vector. Examples of conflicting objectives are performance, reliability and cost. A multicriteria optimization problem consists in finding the ideal solution vector such that the constraints are satisfied and the objective functions are optimal.

These multiple objectives are often in competition with each other, where the improvement of one leads to the deterioration of the others. This conflict between objectives is easily explained: in general, high performance structures tend to have a high cost, while simpler and usually low cost devices will have lower performance. Depending on the constraints, an intermediate solution (satisfactory performance and acceptable cost) may be optimal. In multi-objective problems with conflictual objectives, the optimum is no longer a simple solution as in single-objective problems, but a set of solutions, called the set of best compromises or the Pareto front [DD98].

Among the most important properties of multi-objective optimization problems are the linearity and convexity. The previous properties can impact heavily the efficiency of the methods that can be used to solve the optimization problem. In the following, we recall the definition of a linear and convex optimization problem.

Definition 5.1 (Linear Optimization Problem) *A multi-objective optimization problem is said to be linear if and only if all the following conditions hold:*

- *All objective functions are linear in the parameters (variables) considered in the multi-objective optimization problem*
- *All optimization constraints are linear in the parameters considered in the multi-objective optimization problem i.e., each optimization constraint can be written as a linear combination of the variables that appear in them.*

We said that a multi-objective optimization problem is nonlinear if it is not linear.

Definition 5.2 (Convex Function) *A function $f : R^n \rightarrow R$ is convex if its domain is a convex set and for all $(x_1, \cdots, x_n), (x'_1, \cdots, x'_n)$ in its domain, and all $\lambda \in [0, 1]$, we have*

$$f(\lambda x_1 + (1-\lambda)x'_1, \cdots, \lambda x_n + (1-\lambda)x'_n) \leq \lambda f(x_1, \cdots, x_n) + (1-\lambda) f(x'_1, \cdots, x'_n)$$

Informally, the previous definition means that if we take any two points $(x_1, \cdots, x_n)$ and $(x'_1, \cdots, x'_n)$, then f evaluated at any convex combination of these two points should be no larger than the same convex combination of $f(x_1, \cdots, x_n)$ and $f(x'_1, \cdots, x'_n)$.

Definition 5.3 (Convex Optimization Problem) *A multi-objective optimization problem is said to be convex if and only if all objectives functions are convex functions over convex sets. We said that a multi-objective optimization problem is non-convex if it is not convex.*

5.2.2 Queuing Theory

In this section, we present the fundamental concepts of queues as described in [All90] and we introduce the model that we will use in our solution.

Characteristics of Queuing Models

We speak of a waiting phenomenon whenever certain entities, also called clients, arrive at a system composed of a set of servers in order to receive a service from the latter or wait to be served.

Incoming flow: We are confronted with queuing systems, as soon as we talk about the flow of arriving customers. The arrivals to the queues can be deterministic or random, dependent or independent, individual or grouped, homogeneous or heterogeneous.

Server: The incoming stream is presented to a device, called a server, which provides the requested service. The service time can be described by a probability law or determined from the beginning.

A service discipline: The service discipline is the method of selecting the next client when the server terminates the service of the current client. A number of disciplines are used such as first in first out (FIFO), last in first out (LIFO), prioritized, random, etc.

Queue capacity Queues can have a limited capacity. This means that when it is full, incoming flow will be backed up at the entrance of the system. This factor is important especially. If, for example, a limited-capacity queue is inserted between two consecutive services, when this queue reaches saturation, a blockage will occur at the first service. In the other side, queue can also be considered of infinite capacity.

Kendall created a notation to describe a queue system [Ken53]. The general nomenclature of a queue using this notation is of the form $A/S/c/K/D$ where, A describes the probability distribution of times between customer arrivals, S describes the probability distribution of a customer's service time, c represents the number of servers, K represents the queue capacity, and D represents the service discipline of the queue.

The $M/M/1/\infty/FIFO$ Model

The $M/M/1/\infty/FIFO$ queue is a classic example where clients arrive and are served according to Markov's law, at a single server. The queue capacity of the system is considered to be infinite with FIFO service discipline.

Definition 5.4 (Queue Latency [All90]) *Consider the $M/M/1/\infty/FIFO$ queuing model. Let us denote by λ and μ the mean incoming flow per time period and the mean number of customers served per time period, respectively. The latency $\mathcal{L}$ of the queue i.e., the average time a customer spends waiting in the queue to be served is computed as follows*

$$\mathcal{L} = \frac{\lambda}{\mu(\mu - \lambda)}$$

Note that $\mathcal{L}$ tends to infinity if $\mu \leq \lambda$.

5.3 Adversary model and Problem Statement

In contrast to existing similar approaches, in this work we consider an insider adversary who can control one or more VNFs of the NFV service. Nevertheless, we assume that the compromised VNFs are not known to the entity aiming to deploy the access control

policy. In addition, we assume that the adversary can inject new traffic in the NFV service using the VNFs he/she controls. However, we assume that the traffic transiting through the NFV service can be authenticated. This means that the adversary cannot impersonate other (uncompromised) VNFs by generating and sending network traffic on their behalf. We believe that this last assumption is fairly reasonable since, any access control model is useless if the considered adversary can impersonate other system's entities.

Considering the previous assumptions, traditional solutions for optimal deployment of access control models on virtualized network services become useless and cannot guarantee a correct deployment of the access control policy. When considering an external adversary, an optimal and correct deployment of access control policy requires each rule of the latter to be enforced at least once for deployment correctness and at most once for optimizing the impact related to access control policy enforcement on the NFV service.

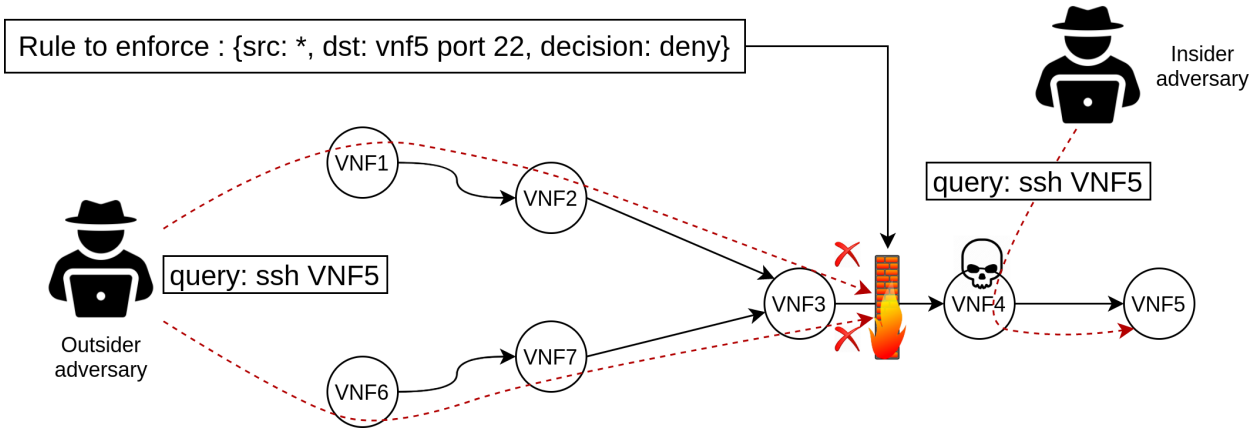


Figure 5.1 – Classic policy deployment strategy in the presence of an insider adversary
© [2022] IEEE

As illustrated in Figure 5.1, when an external adversary is considered, a rule has to be enforced only once for each traffic to prevent it from reaching its destination when violating the rule. However, this is no longer sufficient if an insider adversary is considered. As illustrated, an insider adversary who compromises the VNF4 can easily create traffic that violates the access control policy. Hence, if we consider that the compromised VNFs are not known to the entity aiming to deploy the access control policy, it is necessary to place a firewall at the exit of each VNF to ensure that a compromised VNF cannot inject unauthorized traffic in the network. Nevertheless, when a large NFV

service is considered, a large number of firewalls will be required, which increases the amount of resources and/or may lead to high latency. One solution to optimize the use of the resources of these firewalls consists of making sure that only the policy rules corresponding to the VNFs each firewall is in charge of should be enforced by the latter.

The problem we are tackling in this chapter consists of optimizing the deployment of access control policies for a virtualized network service NFV. We consider an NFV service composed of several VNFs that can be hosted in different physical servers, each having limited available resources e.g., the number of available CPUs and the quantity of available memory. Concretely, resolving our problem consists of finding 1) the optimal number of policy enforcement points (PEP), their corresponding logical (i.e., where to place the PEP in the NFV service) and physical (i.e., in which physical server each PEP is placed) placements, as well as 2) the access control rules that are enforced by each PEP, allowing to reduce both the latency of the NFV service while ensuring a correct deployment of the access control policy. We mainly constrain our problem on physical server resources. As each physical service disposes of a limited resource, it cannot host an infinite number of policy enforcement points.

5.4 System Modelling and Problem Formalization

In this section, we provide a formal modelling of the considered system. We give a formalization of the considered optimization criteria, and we formulate the optimization deployment of access control policies problem. The notations for the variables and the inputs are gathered in Table **??**.

5.4.1 NFV Topology Modelling

NFV services are composed of a set of VNFs $\{n_1, n_2, \cdots, n_m\}$ and a set of forwarding graphs (FGs) which can be seen as oriented graphs. Hence, we model the initial NFV service on which the access control policy is to be deployed by a directed, vertex and edge labeled graph $G = (V, E, L, R)$. The set of vertex V denotes the set of VNFs that compose the NFV service, where v_i denotes the vertex representing the VNF n_i. The set of edges E represents the set of network links between the different VNFs, where $e_{i,j}$ denotes an edge linking the vertices v_i and v_j. The set of labels L denotes

Notation	Description
Inputs	
P	Access Control Policy
FG	Forwarding Graph
G	Oriented Graph
V	Set of VNFs that compose the NFV service
E	Set of network links between the different VNFs
L	Set of physical VNFs locations
R	Set of average flow rates that transit through the set of edges E
s_v	The server in which the VNF v is hosted
S	Set of physical servers
r_e	Average flow rate that traverse the edge e
G	Oriented Graph $G = (V, E, L, R)$
$e_{i,j}$	Link between node n_i and node n_j
s_{mem}	Available memory in the physical server
s_{cpu}	Number of CPUs available on the physical server
d_r	Average delay of single rule
Variables (Parameters)	
$\mathcal{F}$	The set of used policy enforcement points
$\mathcal{F}_s$	The set of used policy enforcement points hosted in the physical server
F_{mem}	Memory used by the set of PEPs hosted in the physical server
F_{cpu}	Number of CPU
E_p	Edge belonging to the path p
$\mathcal{L}_s$	Global latency of service s
$\mathcal{L}_p$	Latency in the forwarding path p
$\mathcal{L}_e$	Latency between two nodes

Table 5.1 – Inputs and Variables Notation © [2022]

the physical VNFs locations, where s_v denotes the physical server in which the VNF represented by the vertex v is hosted. Finally, the set of labels R denotes the set of the averages of flow rates $r_{i,j}$ that transit on an edge $e_{i,j} \in E$.

5.4.2 Policy Deployment

Access control policy is to be deployed on the NFV service through one or several PEPs. In the case in which an insider adversary is considered, and since we are sup-

posing that we do not know which VNF nodes are compromised by the adversary, a correct deployment of the access control policy requires each traffic that transits between two nodes of the NFV service to be authorized by the access control policy. To satisfy the previous condition, a trivial solution would be to enforce the complete policy (i.e., all the rules of the policy) at each network link between two nodes as depicted in Figure 5.2. While it correctly deploys the access control policy, the previous solution leads often to high latency in the NFV service since it is likely to happen that only few rules of the policy will be matched by the traffic, and other rules will be processed uselessly.

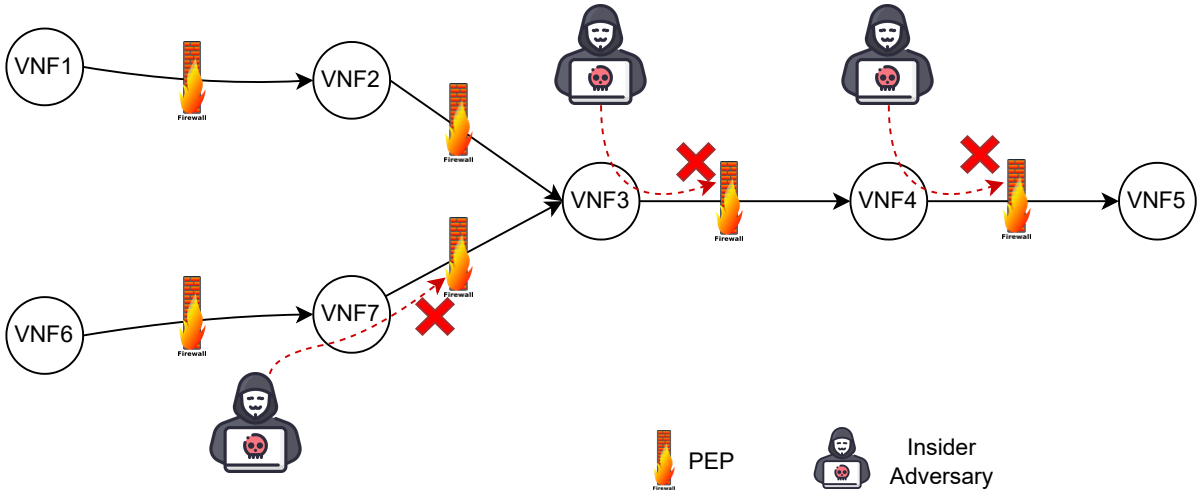


Figure 5.2 – Policy deployment strategy to cope with insider adversaries

To optimize the policy deployment, our solution has to guarantee that, for each link between two nodes, only the rules of the policy that can match traffic that flow through the link are to be enforced. Using the modelling of NFV topology we provide in Section 5.4.1, we formally define the network traffic that can transit through a network link (an edge of G) as follows.

Definition 5.5 *Given a network link represented by an edge $e_{i,j} \in E$ of G. Let us denote by A_{v_i} (resp. D_{v_j}) the set of ancestor (resp. descendant) vertices of v_i (resp. v_j) in G. A network traffic T can flow through $e_{i,j}$ if and only if T originates and has been forwarded by $A_{v_i} \cup \{v_i\}$ or has a destination belonging to $D_{v_j} \cup \{v_j\}$.*

We now define the set of access control rules that should be enforced on each network link (an edge of G). Informally, a rule has to be enforced in a network link if and only if it matches one of traffic that can flow through the link.

Definition 5.6 *Given an access control policy $\mathcal{P}$ and a network link represented by an edge $e_{i,j} \in E$ and the set of traffic $\mathcal{T}$ that can flow through $e_{i,j}$. An access control rule $r = \langle r_{src}, r_{dst}, r_{decision} \rangle$ has to be enforced over e if and only if the following condition holds.*

$$R_{e_{i,j}} = \{r \mid \exists T \in \mathcal{T}, T_{scr} \in r_{src} \text{ and } T_{dst} \in r_{dst}\}_{r \in \mathcal{P}} \tag{5.1}$$

where T_{scr} (resp. r_{scr}) denotes the traffic (resp. the rule) source and T_{dst} (resp. r_{dst}) denotes the traffic (resp. the rule) destination.

Theorem 5.4.1

The policy enforcement strategy described in Definition 5.6 ensures a correct policy deployment in the presence of an insider adversary.

Proof: Let us consider that the access control policy to be deployed is composed of the following set of rules $\{r_1, \cdots, r_n\}$, each rule is denoted as $(r^i_{src}, r^i_{dst}, r^i_{decision})$ where r^i_{src}, r^i_{dst}, and $r^i_{decision}$ represent respectively the subject, the object, and the decision indicating whether it is a permission or denial rule. Let us denote by $T = (T_{src}, T_{dst}, T_a)$ a traffic flowing through the NFV service where T_{src}, T_{dst} and T_a denote respectively the VNF source node of the traffic, the VNF destination node of the traffic, and an authorization result indicating whether the traffic has been authorized or not. A policy is not correctly enforced if one of the following cases hold:

1. $\exists, r_i \in P \; and \; T \in \mathcal{T}$ such that $T_{src} \in r^i_{src}$, $T_{dst} \in r^i_{dst}$, $r^i_{decision} =$ *allow*, and T_a is not authorized.

2. $\exists, r_i \in P \; and \; T \in \mathcal{T}$ such that $T_{src} \in r^i_{src}$, $T_{dst} \in r^i_{dst}$, $r^i_{decision} =$ *deny*, and T_a is authorized.

Informally, case 1 (resp. case 2) happens when a traffic that should (resp. should not) be authorized by the policy is not (resp. is) allowed by the policy enforcement points. In the following, we prove by contradiction that both cases cannot happen.

Case 1 Let us suppose that case 1 holds. This means that for all paths $\mathcal{P}_{\mathcal{T}}$ in the NFV service forwarding graph that links T_{src} and T_{dst}, there exists an edge e in which the

authorization rule r_i is not enforced. Formally, we have

$$\forall path \in \mathcal{P}_{\mathcal{T}}, \exists e \in path \mid r_i \notin R_e.$$

which contradicts Equation (5.1).

Case 2 We now suppose that case 2 holds. This means that there exists a path in the NFV service forwarding graph that links T_{src} and T_{dst} such that, for all edges, the prohibition rule r_i is not enforced. Formally we have,

$$\forall e \in path, \exists path \in \mathcal{P}_{\mathcal{T}} \mid r_i \notin R_e.$$

which again contradicts Equation (5.1). □

Remark 5.4.1

In this section, we propose a new correctly provable policy enforcement model that optimally computes the set of rules of the access control policy that needs to be deployed in each network link linking two VNFs. Since the right rules are deployed in each network link that composes the NFV service, an insider adversary cannot violate the access control policy as long as he/she cannot impersonate other (uncompromised) VNFs, which is the case according to the considered adversary model (Section 5.3). Hence, we answer the research question RQ-5.6

5.5 Latency Quantification

One of the objectives we are considering in this chapter is the minimization of the impact in terms of latency of the deployment of the access control model on the target NFV service. In this section, we provide a formal method for quantifying the latency generated by the deployment of the access control policy on the NFV service.

According to Section 5.4.1, an NFV service is modeled through an oriented graph, which means that a traffic (a flow of network packet) can transit through several possible paths of the oriented graph G. Hence, we define the latency generated by the deployment of the access control policy on the NFV service as the sum of latencies generated at each possible path of the oriented graph representing the NFV service.

Let us denote by $\Phi = \{\phi_1, \cdots, \phi_n\}$ the set of possible paths. Moreover, as we have seen in the previous section, an access control policy is deployed at the network links by a set of PEPs. Hence, the latency generated at each path is in fact the sum of the latencies that are generated at each network link of the path. In view of the previous observations, we formalize the latency generated by the deployment of the access control policy at the NFV service level as follows:

$$\mathcal{L}_s = \sum_{\phi \in \Phi} \sum_{e \in \phi} \mathcal{L}_e. \tag{5.2}$$

where $\mathcal{L}_s$ and $\mathcal{L}_e$ denote respectively the latency generated at the service level and the network link level.

Let us now focus on the latency generated at a network link. As described in Section 5.4.2, our policy deployment strategy requires that a set of rules has to be enforced by a PEP (this is denoted by F) at each network link of the considered NFV service. Therefore, the latency generated at a network link e is equal to the sum of 1) the data transmission delay $\mathcal{L}_{e,F}^{(T)}$ between the two nodes that are connected through the network link and F – the PEP that enforces the access control rules, 2) the rules enforcement delay $\mathcal{L}_F^{(E)}$ which represents the time required by F to enforce the set of rules on a network traffic, and 3) the queuing delay $\mathcal{L}_F^{(Q)}$ at F.

$$\mathcal{L}_e = \mathcal{L}_{e,F}^{(T)} + \mathcal{L}_F^{(E)} + \mathcal{L}_F^{(Q)} \tag{5.3}$$

In the following subsections, we show how to compute the transmission, enforcement, and queuing delays.

5.5.1 Transmission Delay

Given a network link represented by the edge $e_{i,j} \in G$. We define the transmission delay to be the time needed to transfer a network packet from the VNF represented by the vertex v_i and the PEP F added to the time required to transfer the same packet from F to the node VNF represented by the vertex v_j. The transmission delay depends mainly on the physical location of the different involved nodes as illustrated in Figure 5.3. If the considered VNFs are in the same physical server, then the traffic will flow through the virtual network layer. Otherwise, the traffic will flow through the physical network layer.

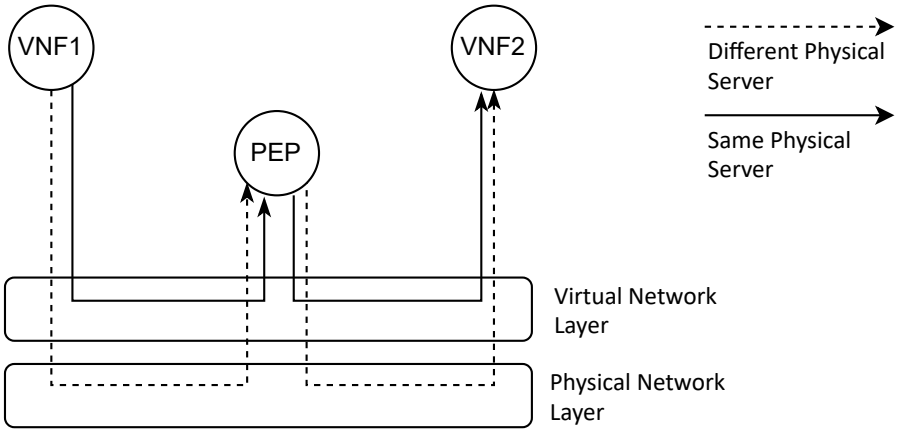


Figure 5.3 – Transmission delay cause by the access control policy enforcement

Let us denote by s_{v_i}, s_{v_j}, and s_F, the physical server hosting the VNFs represented respectively by v_i, v_j, and F.

$$\mathcal{L}^{(T)}_{e_{i,j},F} = \mathcal{D}_{s_{v_i},s_F} + \mathcal{D}_{s_F,s_{v_j}} \tag{5.4}$$

where $\mathcal{D}_{s_i,s_j}$ represents either the average latency between the two physical servers s_i, s_j if $s_i \neq s_j$, or the average latency in the virtual network layer otherwise.

5.5.2 Rules Enforcement Delay

The rules enforcement delay is the time needed by a PEP F to enforce a set of access control rules R_e on a network packet transiting through the edge e. It depends mainly on the number of rules to be enforced and the computational capability (i.e., number of CPUs) of F. Let us denote by d_r the average delay of the enforcement of a single access control rule on a network packet, by F_{cpu} the number of CPUs used by F, and by $||R_e||$ the number of rules that has to be enforced over e. In addition, considering that a PEP can be charged to enforce rules on several edges, we denote by E_F the set of edges on which F has to enforce rule. The rules enforcement delay associated to F is quantified as follows:

$$\mathcal{L}^{(E)}_F = \frac{d_r \cdot \sum\limits_{e \in E_F} ||R_e||}{F_{cpu}} \tag{5.5}$$

5.5.3 Queuing delay

In our model, we suppose that the traffic flowing through a data link (an edge e of G) is processed by a single firewall instance. Consequently, we used the $M/M/1/\infty/FIFO$ presented in Section 5.2.2. The queuing delay associated to a PEP F represents the time a network packet is waiting in F before being processed. According to Definition 5.4, the queue latency is a function of the average traffic rate λ_F flowing through a PEP F and the processing rate μ_F of F. It is formalized as

$$\mathcal{L}_F^{(Q)} = \frac{\lambda_F}{\mu_F(\mu_F - \lambda_F)} \tag{5.6}$$

Let us denote by $\mathcal{I}_e$ the average data rate that transit on the edge e. Considering E_F as the set of edges (network links) in G on which F has to enforce access control rules, the average traffic rate λ_F flowing through F is

$$\lambda_F = \sum_{e \in E_F} \mathcal{I}_e \tag{5.7}$$

In the other side, the processing rate μ_F of F can be expressed as a function of the rules enforcement delay $\mathcal{L}_F^{(E)}$

$$\mu_F = \frac{1}{\mathcal{L}_F^{(E)}} \tag{5.8}$$

Now based on Equations 5.6, 5.7, and 5.8, we have

$$\mathcal{L}_F^{(Q)} = \left(\sum_{e \in E_F} \mathcal{I}_e \right) \cdot \left(\frac{1}{\mathcal{L}_F^{(E)}} \left(\frac{1}{\mathcal{L}_F^{(E)}} - \sum_{e \in E_F} \mathcal{I}_e \right) \right)^{-1} \tag{5.9}$$

Putting all together: Once we have defined how transmission delay, rules enforcement delay, and the queuing delay, we have all the ingredients (Equations 5.2 to 5.9) to give our quantification of the latency generated by the deployment of access control policy at NFV service.

$$\mathcal{L}_s = \sum_{\phi \in \Phi} \sum_{e_{i,j} \in \phi} \left(\mathcal{D}_{s_{v_i}, s_F} + \mathcal{D}_{s_F, s_{v_j}} + \sum_{e \in E_F} \frac{d_r \cdot ||R_e||}{F_{cpu}} + \left(\sum_{e \in E_F} \mathcal{I}_e \right) \cdot \left(\left(\sum_{e \in E_F} \frac{d_r \cdot ||R_e||}{F_{cpu}} \right)^{-1} \cdot \left(\left(\sum_{e \in E_F} \frac{d_r \cdot ||R_e||}{F_{cpu}} \right)^{-1} - \sum_{e \in E_F} \mathcal{I}_e \right) \right)^{-1} \right)$$

5.5.4 Optimization Problem Formulation

The objectives of this work consist of (1) minimizing the impact in terms of latency due to the deployment of an access control policy in an NFV service, 2) minimizing the computational resources that are required by the policy enforcement points. Let $\mathcal{F}$ be the set of PEPs to be used for the policy deployment and $\mathcal{F}_s$ be the set of PEPs hosted in the physical server s. The previous two minimization objectives are represented in Formula 5.10.

$$\text{minimize} \quad \left(\mathcal{L}_s, \sum_{F \in \mathcal{F}} F_{cpu}\right) \tag{5.10}$$

$$\text{subject to} \quad \forall s \in S : \sum_{F \in \mathcal{F}_s} F_{mem} \leq s_{mem} \tag{5.11}$$

$$\forall s \in S : \sum_{F \in \mathcal{F}_s} F_{cpu} \leq s_{cpu} \tag{5.12}$$

Our optimization problem involves a set of constraints to be respected. Constraint 5.11 ensures that, for each physical server, the memory used by the set of PEPs hosted in the physical server cannot exceed the latter's available memory. Similarly, constraint 5.12 ensures that, for each physical server, the number of CPUs to be used by the PEPs hosted in the physical server cannot exceed the number of CPUs available on the physical server.

5.6 Problem Solving

The problem we formalize using the objective function in Formula 5.10 and the constraints in Equations 5.11 and 5.12 is a multi-objective, non-linear, and non-convex (see Propositions 5.6.1 and 5.6.2) optimization problem. The two considered objectives are in conflict with each other. That is, minimizing the used resources leads to increase the latency of the NFV service. Hence, it is not possible to find a single optimal solution that optimizes both considered objectives. Our aim is then to find a set of solutions that are as close as possible to the so-called Pareto-optimal. The Pareto-optimal front is a curve or surface that is formed by solutions providing the best compromise between the objectives.

Proposition 5.6.1: non-linearity

The optimization problem formulated in Section 5.5.4 is non-linear.

The proof of the previous proposition is trivial and, hence, omitted.

Proposition 5.6.2: non-convexity

The optimization problem formulated in Section 5.5.4 is non-convex.

Proof: To prove the non-convexity of the optimization problem formulated in Section 5.5.4, we show that the objective function $\mathcal{L}_s$ is non-convex as following. Let us denote $1/\mathcal{L}_F^{(E)}$ as x and $\sum_{e \in E_F} \mathcal{I}_e$ as y. Based on Definitions 5.2 and 5.3, we show that there exists x_1, x_2, y_1, y_2 and $\theta \in [0, 1]$ such that the following convexity inequality does not hold.

$$f(\theta x_1 + (1-\theta)x_2, \theta y_1 + (1-\theta)y_2) \leq \theta f(x_1, y_1) + (1-\theta) f(x_2, y_2) \tag{5.13}$$

Indeed, it is easy to check that the previous inequality does not hold for $x_1 = 1, x_2 = 3, y_1 = 1, y_2 = 2$ and any $\theta \in [0, 1]$. □

Methods	**Non-linear**	**Non-convex**
Differential evolution (DE) [SP97]	✓	✓
Strength Pareto evolutionary algorithm (SPEA) [ZT98]	✓	✓
Non-dominated sorting genetic algorithm-II (NSGA-II) [SD94]	✓	✓
Niched Pareto genetic algorithm (NPGA) [HNG94]	✓	✗
Multi-objective genetic algorithm (MOGA) [MI$^+$95]	✓	✓
Particle swarm optimization (PSO) [KE95]	✓	✓
Ant colony optimization (ACO) [DB05]	✓	✓
Analytic network process (ANP) [SV13]	✓	✗
Shuffled frog-leaping algorithm (SFLA) [ELP00]	✓	✓
Simulated annealing algorithm (SA) [VLA87]	✓	✓
Plant growth simulation algorithm (PGSA) [WCHW08]	✓	✓
Hungarian algorithm (HA) [DD90]	✓	✗
Mixed-integer nonlinear programming (MINLP) [BP$^+$03]	✓	✗

Table 5.2 – Multi-objective optimization problems comparison regarding the ability to deal with non-linear and non-convex optimization problems

Several approaches have been proposed to solve constrained multi-objective optimization problems. Among these approaches, we found Differential evolution [SP97], Non-dominated sorting genetic algorithm-II [SD94], Simulated annealing algorithm (SA) [VLA87], and Ant colony optimization (ACO) [DB05]. Overall, we find 14 multi-objective optimization methods in the literature. We compare them regarding the ability to deal with non-linear and non-convex optimization problems. Table 5.2 shows the comparison results.

In [AA19], Alothaimeen et al. consider 55 studies published between 2012 and 2019 that use the methods presented in Table 5.2 to solve Multi-objective optimization problems in different domains. These optimization methods were used to solve different numbers of objectives (ranged between 2 and 7) at a time. Figure 5.4 reports the frequency usage of each method to solve the considered Multi-objective optimization problems. Note that we consider only the methods that can deal with both non-linear and non-convex multi-objective optimization problems.

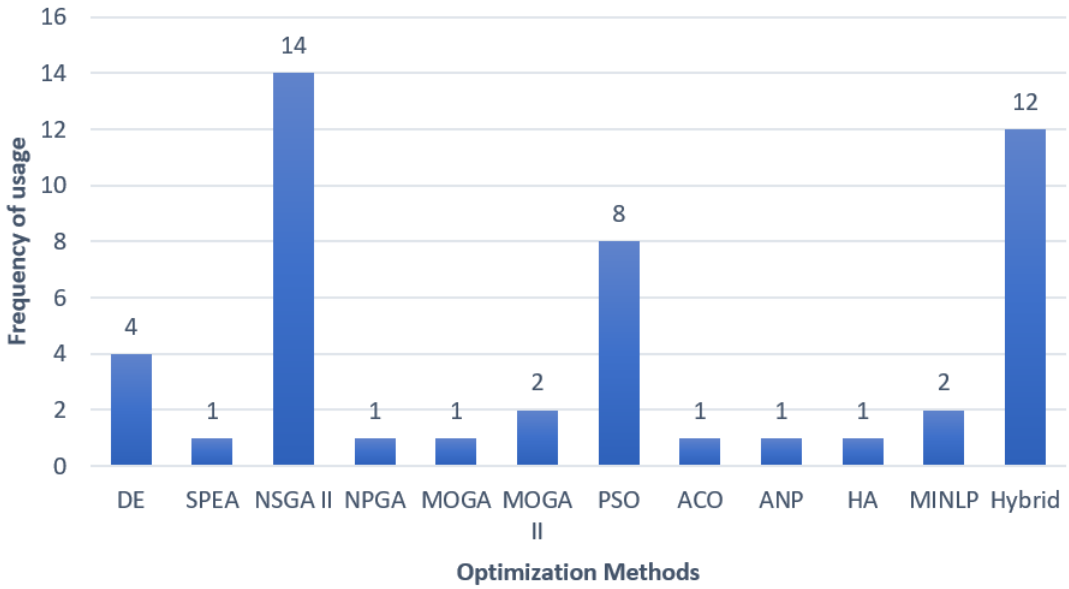


Figure 5.4 – The frequency of usage of the different optimization methods in the considered studies. The "Hybrid" method pairs two or more optimization methods for solving the considered problem.

In view of the above, we can emphasize that among the multi-objective methods used in the literature, NSGA-II was the most used to solve the optimization problems of the considered studies. NSGA-II has proven its capability in solving optimization prob-

lems in different domains. In addition to its popularity among researchers, NSGA-II has many advantages that make it suitable for many types of optimization problems such as obtaining diverse solutions in Pareto-front, low computational complexity, solving problems that involve non-linearity, non-convexity, and non-differential problems, as well as the support of parallel computation.

Deb et al. [DPAM02] proposed an enhancement of the NSGA algorithm, the NSGA-II, which uses a faster (than NSGA) non-dominance based sorting procedure and uses a comparison operator based on a crowding calculation to improve diversity of solutions. In this work, we use an improvement of the NSGA II [WSZ+18] for solving the problem we described in Section 5.5.4.

5.6.1 NSGA II

The NSGA-II algorithm is based on a multi-level classification of individuals. This last one uses a faster sorting procedure than its predecessor (NSGA [SD94]), based on non-dominance or Pareto optimal, an elitist approach which allows to preserve the diversity of the populations, by saving the best solutions found during the previous generations on the one hand, and on the other hand a comparison operator based on a Crowding distance calculation. The working principle of the algorithm is detailed in Figure 5.5.

Population initialization In this step, the population is initialized based on the considered optimization problem parameters such as, number of variables, number of constraints. Particularly, for sake of efficiency and effectiveness, the initial population should satisfy the constraints considered in the optimization problem. For instance, in our optimization problem, the population should be initialized such that the two optimization constraints in Formulas 5.11 and 5.12 are satisfied.

Non-dominance sorting The current population is sorted based on non-dominance. That is, an individual is said to dominate another if the objective functions values associated to it is no worse than the- values associated to the objective functions for the other, and at least in one of its objective functions value the one associated to the other. The sort algorithm [DPAM02] is described in Algorithm 4.

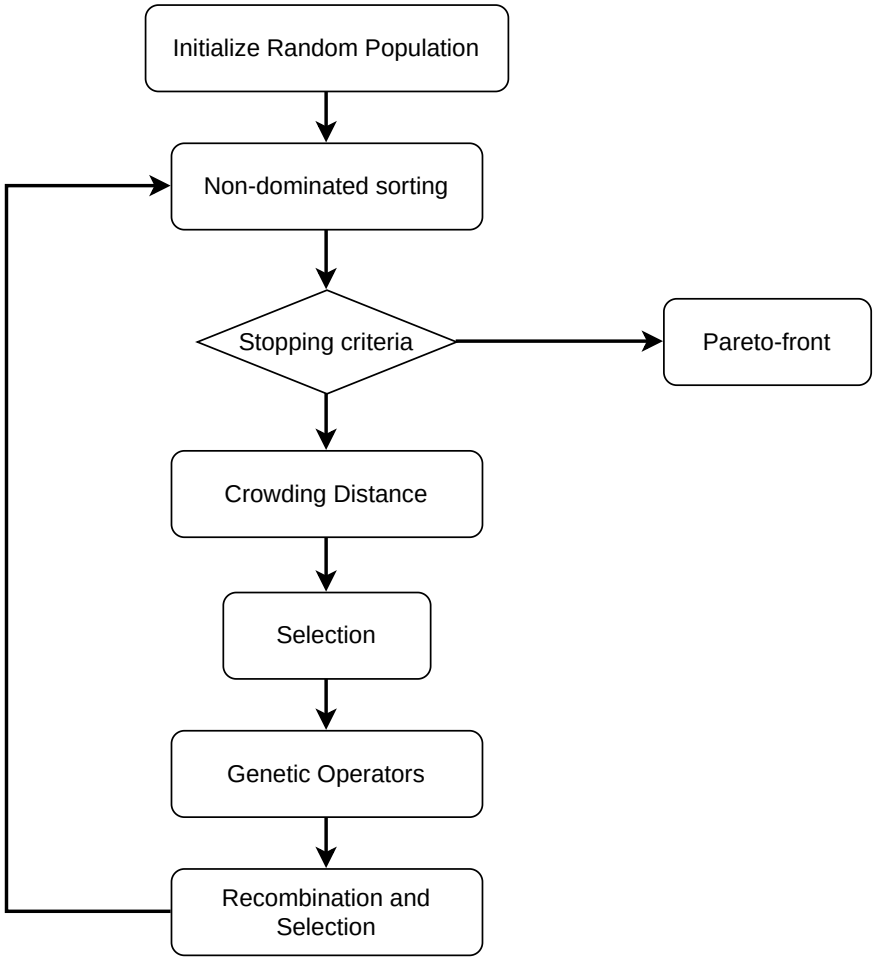


Figure 5.5 – NSGA-II algorithm

Crowding Distance To manage diversity at the level of Pareto sets, the algorithm uses a very particular implementation of the selection operator, the crowded tournament. The latter relies on the crowing distance which represents the euclidean distance between each individual in a front based on the considered n objectives in the n dimensional hyper space. In the Crowded distance assignment procedure, for each solution, we evaluate the density of the solutions present around it. Assigning this value will have the effect of decreasing the chances of survival of a solution present in a region where several other solutions are concentrated. Concretely, as shown in Figure 5.6, assigning a large distance to the first and last solutions, for each objective function, gives priority

to the extreme solutions of a Pareto set.

```
   Input: 𝒫    /* the population to sort */
 1 foreach p ∈ 𝒫 do
 2 |   S_p = ∅   /* the individuals that is being dominated by p */
 3 |   n_p = 0    /* the number of individuals that dominate p */
 4 end
 5 foreach p ∈ 𝒫 do
 6 |   foreach q ∈ 𝒫 do
 7 |   |   if p dominates q then
 8 |   |   |   S_p = S_p ∪ {q};   n_q = n_q + 1
 9 |   |   end
10 |   end
11 |   if n_p = 0 then
12 |   |   p_rank = 1;   ℱ_1 = ℱ_1 ∪ {p}
13 |   end
14 end
15 i = 1
16 while ℱ_i ≠ ∅ do
17 |   Q = ∅    /* the members of the next front */
18 |   foreach p ∈ ℱ_i do
19 |   |   foreach q ∈ S_p do
20 |   |   |   n_q = n_q − 1
21 |   |   |   if n_q = 0 then
22 |   |   |   |   q_rank = i + 1    /* q belongs to the next front */
23 |   |   |   |   Q = Q ∪ q
24 |   |   |   end
25 |   |   end
26 |   end
27 |   i = i + 1;   ℱ_i = Q
28 end
```

Algorithm 4: NSGA II: Non-dominance sorting

For the intermediate solutions, thanks to the sorting vector, the distance is given by the half perimeter of the cuboid surrounding these solutions. In the example shown in Figure 5.6, the solution z_2 is the one that is more likely to be rejected since the

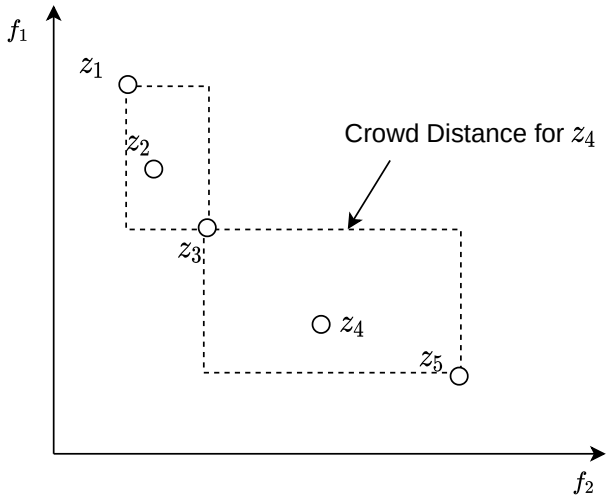


Figure 5.6 – Hypothetical case of the Crowded distance assignment

perimeter of its cuboid is the smallest.

Selection Once the individuals are sorted on the basis of non-domination and the crowding distance is assigned, the selection is performed using the crowded comparison operator ($\prec$) that is defined as follows.

Definition 5.7 (crowded comparison operator) *Given two individuals p and q, $p \prec q$ holds if one of the following conditions holds:*

- $p_{rank} < q_{rank}$
- *If p and q belong to the same front, the crowd distance assigned to p is greater than the one assigned to q*

The individuals are then selected by performing a binary selection with crowded comparison operator.

Genetic Operators NSGA II relies on two generic operations: simulated binary crossover (SBX) [BD01] and polynomial mutation [Kak04].

Recombination and Selection In this step, the NSGA II algorithm combines the offspring population with the current generation population and selects the individuals

that will compose the next generation. As all the previous and current best individuals are part of selected population, elitism is ensured. Since the population is sorted on the basis of non-domination, the new generation is filled by each subsequent front until the population size exceeds the size of the current population. If, by adding all the individuals from the front $\mathcal{F}_j$, the population exceeds N, then the individuals from $\mathcal{F}_j$ are selected on the basis of their crowding distance in descending order until the population size is reachesn N.

Remark 5.6.1

In Sections 5.5, we propose a model to quantify the impact in terms of latency of the deployment of the access control model on the target NFV service. Then, we define a minimization problem that aims to minimize both the impact in terms of latency in a NFV service and the computation resources that are required by the policy enforcement points. In Section 5.6, we show that the proposed optimization problem is nonlinear and non-convex. Then, based on these characteristics, we choose the adapted optimization algorithm. Hence, we answer the research question RQ-5.7.

5.7 Implementation and Simulation

In this section, we aim to evaluate our proposed optimal access control policy deployment model. To meet the previous objectives, we built the simulation framework represented in Figure 4.9. The major functional components are described in the following.

Virtual Network Simulator This module is responsible of the creation of the NFV service topology and the simulation of the impact in terms of latency resulting from the deployment of an access control policy over the created NFV service topology. It is composed of the following sub-modules:

- Random NFV topology generator (RTG): This module provides a random generation of NFV service topology capability. It allows to create an NFV service topology by randomly selecting, the number of VNFs, the number of paths that composes the NFV forwarding graph, the VNFs that compose each path, the

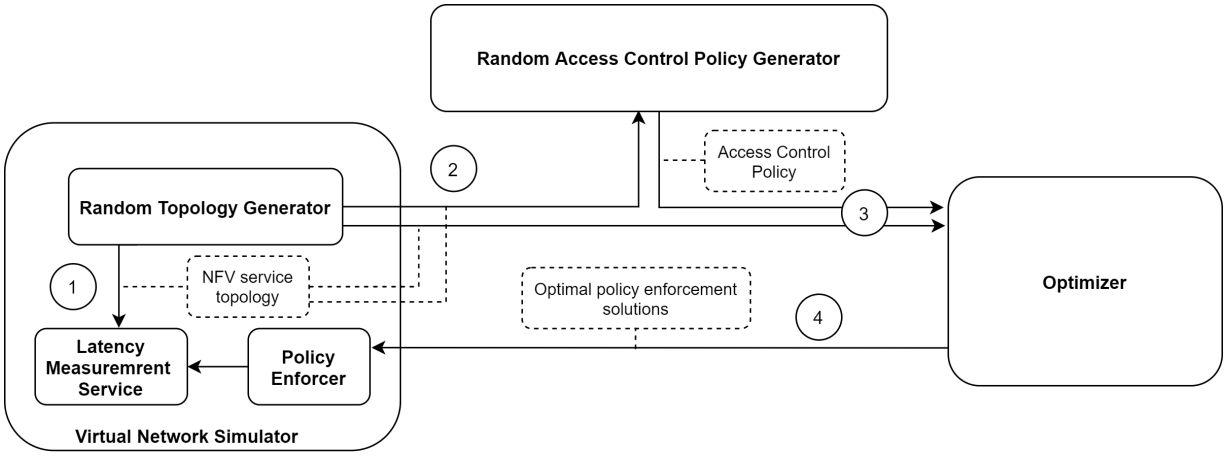


Figure 5.7 – The used simulation framework

number of physical servers that hosts the VNFs, and the physical location of each VNF.

- Latency measurement service (LMS): This module is responsible of measuring the latency at the NFV service topology. The method used by the service to measure the latency is described as following. Using the target NFV service topology, the LMS installs a User Datagram Protocol (UDP) client in the first VNF node of each path of the forwarding graph and installs an UDP server in the last node of each path. Then, it creates randomly generated UDP traffic that flow through the different paths that compose the NFV service topology. Meanwhile, the LMS observes the sending and receiving times of each packet in order to compute the average latency of each path. The overall latency of the NFV service is computed by summing up the average latencies of all the paths that compose the service topology.

- Policy Enforcer (PE): This sub-module is responsible of the deployment of the access control policy enforcement solutions that are returned by the Optimizer. For each enforcement solution, the PE places the enforcement points (i.e., firewalls) according to their placement in the enforcement solution, then informs the LMS to measure the impact in terms of latency of the deployed enforcement solution.

As a virtual network simulator, we used NS3 [ns3]. The RTG, LMS, and PE sub-modules are developed in C++ and used as NS3 modules.

Random policy generator (RPG) This module is responsible of generating the random access control policies to be deployed on a target NFV service. To generate the policies, the RPG builds access control rules by choosing, for each rule:

- Two random, yet connected VNFs from the topology of the target NFV service. The first represents the subject of the rule while the second is considered as the object of the rule.
- A random decision (allow or deny). We have implemented the RPG in python. Two VNF nodes are said to be connected if there is a path in the NFV service topology that allows a network traffic to flow from a node to the other.

Optimizer This module takes as inputs the target NFV service topology and the access control policy to be deployed. Then it processes them to find the optimal solutions for the deployment of the access control policy. A deployment solution includes the locations of the firewalls in the physical servers, the set of links that are to be processed by each firewall and the computational resources (i.e., the number of CPUs) that are allocated for each firewall. We use the Python based implementation of NSGA II provided in jMetalPy framework [BHNGN+19].

We now focus on the simulation flow and the interactions between the different modules. As first step, we use the NS3's RTG module to generate a random NFV service topology by choosing randomly the number of VNFs, the number of paths inside the forwarding graph, the number of VNFs inside each path, and the number of physical servers that host the NFV service. In the second step, the generated topology will be sent to the LMS to measure the initial latency i.e., the average latency observed in NFV service before the deployment of the access control policy. Then, in the third step, the generated service will be sent to the RPG in order to generate random access control rules that are going to be deployed on the NFV service. In the fourth step, the generated topology and the access control policy will be sent to the Optimizer. Once the latter receives these elements, it uses the NSGA II algorithm to find the optimal policy deployment solutions. The latters are sent back to the PE module which enforces the solutions one by one on the NS3 topology of the NFV service by placing the policy enforcement points i.e., firewalls in their locations and deploys the set of access control rules that should be enforced at each enforcement point. For sake of simplicity, we simulate the traffic processing at firewall level by adding a delay that represents the

time needed by the firewall to process the traffic according to the number of rules to enforce and the number of CPUs it can use. The added delay is computed as described in Equation 5.5.

Afterwards, the PE notifies the LMS to remeasure the latency inside the NFV service. Concretely, the latter is computed as the sum of the latencies measured in each path in the NFV service (see Equation 5.2). The latency in each path is measured by computing the time needed by a network packet to go all the way through the path. Finally, the LMS computes the impact of the deployment of the access control policy as the difference between the initial latency and the one observed after the deployment of the policy on the NFV service.

We experimentally evaluate the performance of our optimal access control policy enforcement model. All the conducted evaluations were performed in a server running Linux with an Intel XeonE5-2680 v4 Processor with 32 vCPU and 128 GB of RAM. In our experimentation, we aim to evaluate the impact of the deployment of the access control policy according to 1) the size of the NFV service, 2) the size of the access control policy to be deployed, 3) the number of physical servers on which the NFV service is hosted, 4) and the size of the traffic that flows through the NFV service.

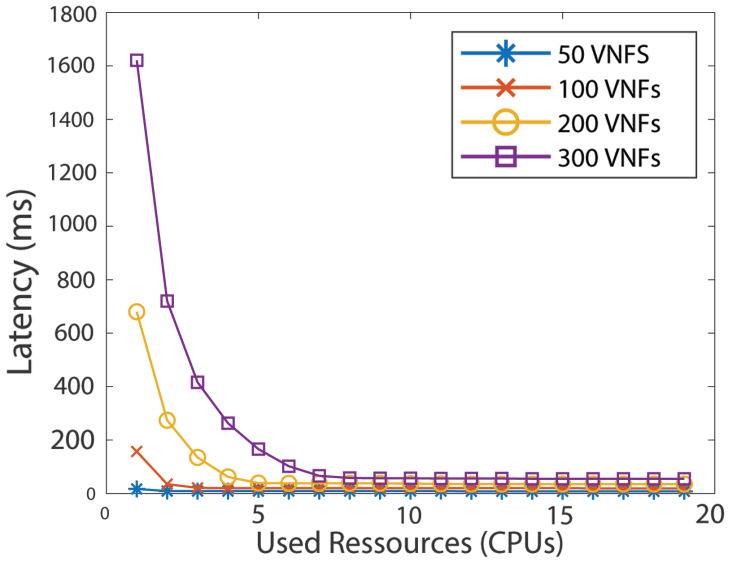


Figure 5.8 – Optimal trade-offs between the impact in terms of latency of the deployment of the access control policy and the needed resources as a function of the number of VNFs that composes the NFV service. © [2022] IEEE.

To evaluate the impact of the deployment of the access control policy according to

the size of the NFV service i.e., the number of VNFs that composes the NFV service, we consider the configuration in which all VNFs that compose the NFV service are hosted in a single physical server. We also consider an access control policy composed of 100 rules. In this evaluation, we vary the number of nodes to study its impact on the trade-offs between the latency resulting from the deployment of the access control policy and the needed resources. Figure 5.8 reports the obtained result. As we can see, the latency grows exponentially as the number of VNFs in the service increases and the used resources decrease. In addition, the results show, for each considered size of NFV service, the resources required to achieve a near optimal latency. For example, when an NFV service composed of 300 VNFs is considered, the policy enforcement points need to use 8 CPUs to reduce to the best the latency.

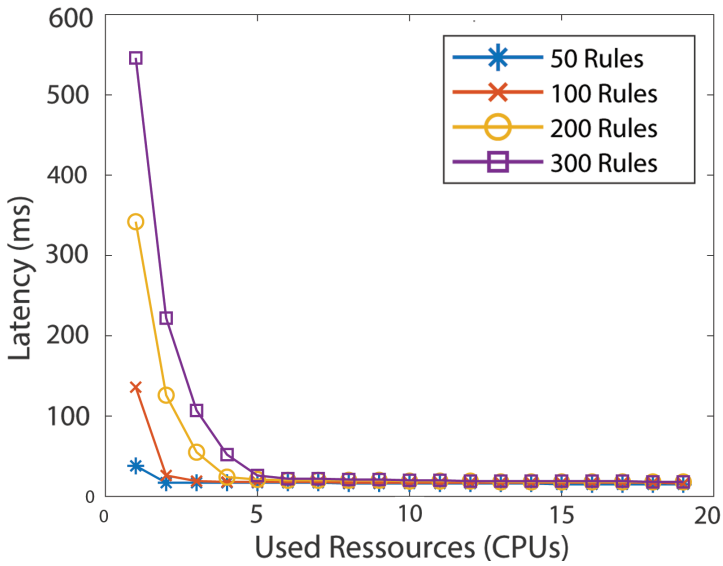


Figure 5.9 – Optimal trade-offs between the impact in terms of latency of the deployment of the access control policy and the needed resources as a function of the number of rules in the policy to be deployed. © [2022] IEEE.

We now evaluate the impact of the deployment of the access control policy according to the number of rules in the policy. We consider a configuration in which an NFV service composed of 100 VNFs is hosted in the same physical server. We randomly generate access control policies with different sizes: 50, 100, 200, and 300 rules. Then, we perform the simulation to see how the trade-offs between the latency resulting from the deployment of the access control policy and the needed resources change. As illustrated in Figure 5.9, the latency is growing exponentially as the number of rules in-

creases and the used resources decrease. The simulation results show the resources required to achieve a near optimal latency when deploying each of the considered policies. For example, a policy composed of 300 rules requires the usage of 5 CPUs to achieve a near-optimal latency.

Policy enforcement points' physical location are extremely important when an NFV service hosted on several physical server is considered. The goal of this evaluation is to study the impact of the deployment of the access control policy according to the number of physical servers in which the NFV service is hosted. Hence, in this configuration, we consider an NFV service composed of 100 VNFs and an access control policy composed of 100 rules. The result of the simulation is depicted in Figure 5.10.

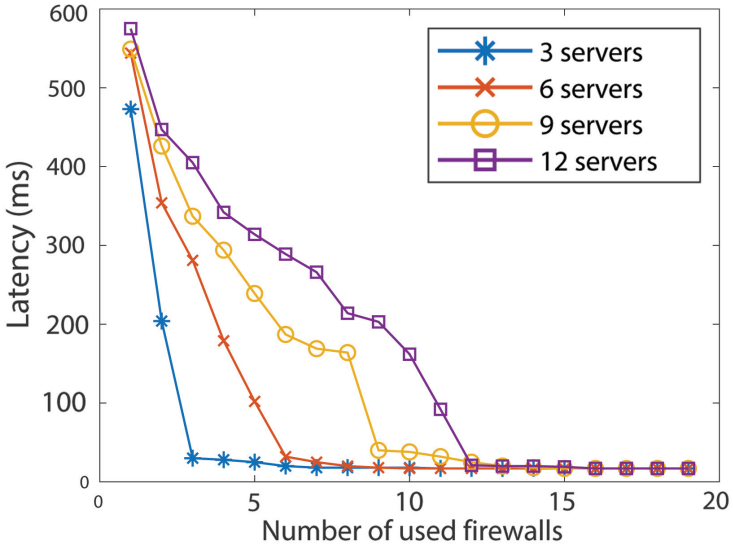


Figure 5.10 – Optimal trade-offs between the impact in terms of latency of the deployment of the access control policy and the needed number of firewalls as a function of the number of physical servers. © [2022] IEEE.

Similarly to previous simulations, the latency grows exponentially as the number of used policy enforcement points (firewalls) decreases. This mainly due to inter physical server's transmission delay since, due to the limited number of firewalls, large part of the traffic flowing through VNFs have to be sent to a firewall hosted in different physical server. The results show also that once the number of used firewalls equals the number of physical servers, the observed latency falls to a near-optimal value.

Seeking to have better understanding of the result obtained in the previous experiment, we compute the different observed delays resulting from the deployment of an

access control policy composed of 100 rules on an NFV service composed of 100 VNFs hosted in 12 different physical servers. The results are shown in Figure 5.11.

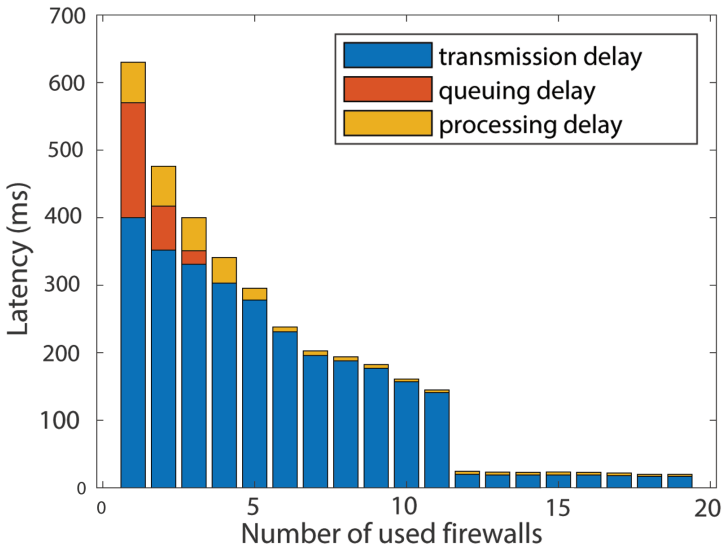


Figure 5.11 – Different observed delays resulting from the deployment of an access control policy composed of 100 rules on an NFV service composed of 100 VNFs hosted in 12 different physical servers. © [2022] IEEE

The results show that the transmission delay (the delay between the VNF sending the traffic and the firewall) represents the most part of the observed latency. The reason behind that is the relatively high delay between physical servers compared to the delay inside the same server. Hence, when the number of firewalls to be used is less than the number of servers, it is not possible to place a firewall in each physical server. Hence, there is often a need to transmit the traffic to a firewall that is placed in a different physical server, which considerably increases the transmission delay due to the high inter-server latency. However, as soon as the number of firewalls equals the number of physical servers, it is possible to place a firewall in each physical server resulting in the elimination of the inter-server latency.

Finally, we evaluate the time required for the optimizer to find the optimal policy enforcement solution to be deployed. Figure 5.12 reports the average time needed for the optimized implementing the genetic algorithm NSGA II to solve our multi-objective optimization problem (Formulas 5.10-5.12) as a function of the number of VNFs that compose the target service and the number of rules that needs to be enforced.

According to the previous Figure, the optimization algorithm used in our solution is

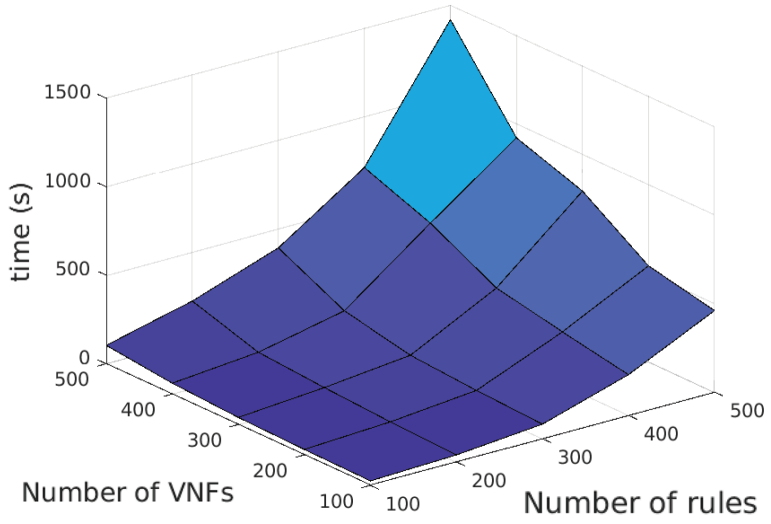


Figure 5.12 – The time required for the optimizer to find the optimal policy enforcement solution to be deployed. © [2022] IEEE.

quite efficient when medium size policy (i.e., number of rules less or equal than 200) is to be enforced on medium size (i.e., number of VNFs less or equal than 200) NFV service. However, when a large number of rules (e.g. 500 rules) are to be enforced on a quite complex NFV service (composed of 500 VNFs), our approach based on NSGA II algorithm takes around 23 minutes to find the optimal deployment of the access control rules. While this may be an acceptable delay when launching the NFV service, we believe that this might be too much for specific use case when we should have an immediate application of the policy updates. One possible solution to reduce this delay is to use our graph-based modelling of the topology and the policy deployment problem to exclude all the elements (e.g., the nodes and the edges of the graph) of the NFV service that will not be impacted by the policy updates. The multi-objective optimization problem will then include only the graph nodes and edges that will be impacted by the policy updates which can drastically reduce the time needed by the NSGA II for solving the optimization problem. We leave the investigation of this solution to future work.

5.8 Conclusion

In this chapter, we present an approach allowing a correct and optimal deployment of access control policies on NFV services in the presence of insider adversaries. We firstly propose a formal modelling of the optimal deployment of access control policy problem that allows to model and quantify the resources consumed and the impact in terms of latency that are to be generated by the access control policies deployment. We formally show that the proposed model ensures a correct enforcement of access control policies in the presence of an insider adversary who can control one or more unknown nodes (VNFs) that compose the NFV service. To the best of our knowledge, the solution we propose in this chapter is the first that can provide simultaneously all the aforementioned properties. Finally, we build a simulation framework to evaluate the effectiveness of the proposed optimization method.

The conducted experiments clearly show that the model we propose in this chapter allows security administrators to have enough information e.g., the introduced latency according to the used resources and the introduced latency according to the number of PEPs, to make decision on the right trade-off between the quality of service degradation and the used resources that should be considered.

In respect to this dissertation main research questions, the approach we propose in this chapter fully answers the questions of "How to improve previously proposed solutions to deal with insider adversaries?" (RQ-5.6) and the question of "How to enforce access control policy while minimizing both the consumed resources and the impact caused by the access control policy deployment?" (RQ-5.7).

CHAPTER 6

CONCLUSIONS AND PERSPECTIVES

Contents

In conclusion, we provide an overview of how the different research objectives presented in the introduction were followed and the different contributions that resulted. We then reflect on how our contributions can be improved and provide new research directions.

6.1 Conclusion

The main research goal of this dissertation was to improve the security of NFV services through access control policies enforcement. More precisely, to improve the security of NFV services by defining a framework allowing a correct, efficient, and optimal deployment of access control requirements.

Chapter 2, exploratory by nature, scrutinized the security of NFV-based networks. First, it proposes an extensive literature review of the existing and potential security issues and threats in the NFV infrastructure. Then, to provide a better understanding of the impact of these threats, we classify the issues and threats according to the components that are affected by them. Second, we review existing solutions for enforcing access control policies in NFV infrastructures. Third, we provide a comprehensive comparative overview of existing solutions with respect to several properties, including the

considered adversary model, effectiveness, correctness, optimality, etc. The comparative overview illustrates the main improvements that need to be made to allow correct, efficient, and optimal deployment of access control requirements on NFV service.

In Chapter 3, we address fundamental questions about the design of an access control policy enforcement in NFV service framework that is simultaneously highly expressive, correct, efficient, and easily deployable. First, we define an expressive specification model to be able to express high-level access control requirements to be enforced over network services. Then, we show that our specification model can correctly express access control requirements specified using well known access control models such as RBAC, ABAC, and ORBAC. Second, we propose a provably correct method for refining high-level access control requirements towards a Domain Type Enforcement (DTE) concrete specification. Finally, our model defines an ETSI-NFV compliant and efficient enforcement method, as illustrated by the different conducted experimental evaluations in Section 3.4.

Compared to existing models, our framework offers several advantages to VNF users. First, it is generic in the sense that our model allows to handle the deployment of policies expressed using most well known access control models such as RBAC, ORBAC, ABAC. Second, it complies with the ETSI-NFV infrastructure because it does not require any modification of the latter for policy deployment. The conducted experimentation shows that the implementation of the proposed model is quite efficient. The deployment of a security policy composed of 500 rules introduces less than 2 ms delay for the round-trip time of a network packet. Finally, our policy enforcement method is scalable as it is possible to add as many enforcement points as needed (e.g., for load balancing purposes) without impacting the functioning of the network services.

The policy enforcement framework proposed in Chapter 3 was mainly designed to enforce closed access control policies. Unfortunately, the latter are known to be suffering from a lack of expressiveness when rules containing exceptions need to be enforced. In contrast, mixed policies are well adapted to express high-level access requirements containing exceptions. Nevertheless, it has been shown in [ACCB07] that mixed high-level access control policies containing exceptions often lead to quite complex concrete configurations. In Chapter 4, we investigate the deployment of high-level mixed and complex access control policies containing exceptions using the framework proposed in Chapter 3. First, we propose an approach that straightforwardly transforms a high-level mixed access control policy towards a closed policy that can be

enforced using the framework proposed in Chapter 3. Then, we show theoretically that this approach is quite unscalable as we prove that the number of rules in the closed and exception-free policy grows exponentially in the number of exceptions in the high-level access control policy to be deployed. Hence, to overcome the previous scalability problem, as a second step, we extend our access control policy enforcement framework by proposing a priority-based DTE model. The latter enables a clean and efficient deployment of complex access control policies containing exceptions and / or conflict rules on NFV services. Specifically, the proposed priority-based DTE model does not introduce additional low-level rules compared to the high level security policy to be deployed, which allows security administrator to straightforwardly understand and update the deployed concrete level policy. Our model relies on a provably correct approach for exception management in DTE specification. The conducted empirical evaluations show that the priority-based DTE model we are proposing improves drastically the efficiency of the enforcement of big and complex policies that contain exceptions.

The last part of this dissertation (Chapter 5) improves our access control policy enforcement in NFV service framework in two ways. First, considering the facts that (1) NFV services, as any other virtualized infrastructure, can be compromised and partially controlled by an adversary and that (2) the solutions we propose in Chapters 3 and 4 consider only outsider adversaries that are aiming to remotely bypass the enforced access control policy. In the first part of Chapter 5, we consider a strong adversary model in which we assume that we are dealing with an insider adversary who can control one or more unknown nodes (VNFs) that compose the NFV service. Then, we propose a new correctly provable access control policy enforcement model that optimally computes the right set of rules of the access control policy that needs to be deployed in each network link linking two VNFs. Second, considering the facts that (1) the enforcement of access control policy requires resources in terms of computation and storage, (2) often impacts the functionality provided by the target NFV service i.e., by introducing latency due to the traffic analysis and rule enforcement. In the second part of Chapter 5, we propose a model to quantify the impact in terms of latency caused by the deployment of the access control model on the target NFV service. Then, we define a minimization problem that aims to minimize both the impact in terms of latency in a NFV service and the computation and storage resources that are required by the access control policy enforcement points. Afterwards, we show that the proposed optimization problem is nonlinear and non-convex and we use an improve-

ment of the Non-dominated Sorting Genetic Algorithm NSGA II for solving it. Finally, we build a simulation framework to evaluate the effectiveness of the proposed optimization model. The conducted experiments clearly show that the optimization model we propose allows security administrators to have clear and enough information e.g., the introduced latency according to the used resources and the introduced latency according to the number of PEPs, to make decision on the right trade-off between the quality of service degradation and the quantity of resources that should be dedicated to the access control policy enforcement points.

We conclude this subsection with a summary of the research questions that have been investigated in this dissertation and the chapters in which they were addressed (Table 6.1).

6.2 Perspectives

In this section, we describe some future research directions that we identified during this dissertation.

6.2.1 Dynamic deployment of access control requirements

After their enforcement, access control policies are often subject to update to allow adding/revoking specific accesses. As we have seen in Chapter 5, the optimization of the resources required for policy enforcement and the impact on the quality of service requires several minutes when dealing with complex NFV service (500 VNFs). While this delay can be tolerated during the target NFV service setup, it is too much when specific access control requirements should be updated and instantly enforced.

The access control policy enforcement framework proposed in this dissertation can be extended to support optimal yet dynamic enforcement of access control requirements on NFV services. One possible solution to meet the previous objective consists of reducing the optimization delay by using our graph-based modeling of the NFV service topology and the policy deployment problem to exclude all the elements (e.g., the nodes and the edges of the graph) of the NFV service that will not be impacted by the policy updates. The multi-objective optimization problem will then include only the graph nodes and edges that will be impacted by the policy updates which can drastically reduce the time needed by the NSGA II for solving the optimization problem.

Research Question	Chapter/Section
RQ-1 What are the security threats in NFV infrastructure?	Sec. 2.3
RQ-2 What are the security threats that can be addressed/reduced by Access control policy enforcement?	Sec. 2.5.1
RQ-3 What is the state of the art with respect to the access control policy enforcement in virtualized network infrastructure?	Sec. 2.5
RQ-4 What are the relevant properties that are satisfied/unsatisfied in existing solutions?	Sec. 2.5.4
RQ-5 How to create *efficient*, *high expressive*, *correct*, and *easy deployable* approach allowing to *optimally deploy mixed* and *complex* access control policies?	Ch. 3 – 5
RQ-5.1 How to define a sufficiently expressive high-level specification model that allows to correctly express policies specified using other well known access control models such as RBAC, ABAC, and ORBAC?	Sec. 3.3
RQ-5.2 How to correctly refine the high level access control policy towards concrete-level deployable requirements?	Sec. 3.3
RQ-5.3 How to define an ETSI-NFV compliant deployment of the concrete-level requirements?	Sec. 3.3 – 3.4
RQ-5.4 Can we use the access control policy deployment solution proposed in Chapter 3 to efficiently deploy high-level mixed and complex access control policies containing exceptions?	Sec. 4.3
RQ-5.5 how can we improve the proposed solution to deal efficiently with high-level mixed and complex access control policies containing exception?	Sec. 4.4
RQ-5.6 How to improve previously proposed solutions to deal with insider adversaries?	Sec. 5.4
RQ-5.7 How to extend our access control policy enforcement framework to minimize both the consumed resources and the impact caused by the access control policy deployment?	Sec. 5.5 – 5.7

Table 6.1 – Summary of the research questions that have been investigated and the chapters in which they were addressed.

6.2.2 Heterogeneous security requirements enforcement on NFV services

By analyzing some real-life scenarios of NFV services, we realize that the security and utility (functionality) requirements specified by NFV service owners are quite different in each scenario. Moreover, they are often heterogeneous (e.g., access control requirements, confidentiality requirements, privacy requirements, etc.). Security mechanisms allowing to provide those security requirements have been the focus of huge interest. These mechanisms are known to be effective when used independently. However, in many situations, they must be combined appropriately to provide security functionality without one interfering with the other.

The framework proposed in this dissertation can be extended by defining a sufficiently expressive language to specify heterogeneous security requirements that take into account the topology of the target NFV service as well as the data that needs to be processed. Moreover, it is necessary to develop a formal theoretical means to analyze the consistency of these requirements. One possible approach is to extend the logical formalism of the proposed framework to formally represent security requirements and their enforcement in the target system.

As a second objective, it is therefore necessary to propose a richer formalism to formally specify the requirements that each security mechanism can guarantee. The approach is to formally characterize the security requirements guaranteed by the considered security mechanisms by relying on a logic-based language. The objective is to have a unique formal framework to reason about the security needs expressed in the policy and the requirements guaranteed by the available security mechanisms.

The innovative principle is therefore to consider that all the available security mechanisms constitute a "toolbox" from which the security architect can "pick and choose" to meet specific security needs. In this context, how can one verify that the various mechanisms selected actually guarantee the requirements specified in the security policy? One possible approach to answer the previous question is to rely on the formal specification of the security policy and the security mechanisms to determine the most appropriate mechanisms to meet the data outsourcing and security needs expressed in the policy.

APPENDIX A

PUBLICATIONS

The following are the publications during the course of my thesis:

1. Smine, M., Espes, D., & Pahl, M. O. (2022, April). Optimal Access Control Deployment in Network Function Virtualization. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium (pp. 1-9). IEEE.

2. Smine, M., Espes, D., Cuppens-Boulahia, N., Cuppens, F., & Pahl, M. O. (2020, December). A priority-based domain type enforcement for exception management. In International Symposium on Foundations and Practice of Security (pp. 65-81). Springer, Cham.

3. Smine, M., Espes, D., Cuppens-Boulahia, N., & Cuppens, F. (2020, June). Network functions virtualization access control as a service. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 100-117). Springer, Cham.

4. Smine, M., Cuppens, N., & Cuppens, F. (2018, October). Effectiveness and Impact Measurements of a Diversification Based Moving Target Defense. In International Conference on Risks and Security of Internet and Systems (pp. 158-171). Springer, Cham.

APPENDIX B

RÉSUMÉ EN FRANÇAIS

B.1 Contexte et motivation

Pour répondre rapidement et efficacement à la croissance des besoins de services réseaux nouveaux et diversifiés, les réseaux d'aujourd'hui doivent être flexibles, adaptables, réactifs, robustes et dotés de ressources importantes (capacité de traitement, stockage et bande passante) [MS14]. Le modèle selon lequel sont construits les réseaux des générations précédentes, qui repose sur du matériel propriétaire souvent fortement couplé à une technologie particulière (IPv4 par exemple), n'est pas en mesure de répondre efficacement et rapidement à ces besoins pour réaliser cette transformation. C'est donc de ce constat quiest né le paradigme de la virtualisation des fonctions réseau (NFV). Avec la NFV, les fonctions réseau telles que le stockage, le calcul, le filtrage et le NAT, peuvent être découplées du matériel, ce qui permet de les virtualiser en Fonction de réseau virtuel (VNF) à installer n'importe où sur l'infrastructure réseau, aussi bien physiquement que virtuellement. L'objectif est donc de simplifier la gestion du cycle de vie des services réseau, d'optimiser les ressources et d'améliorer l'efficacité opérationnelle et de permettre le développement rapide de nouveaux services réseau tout en maximisant la flexibilité pour l'évolutivité et l'automatisation.

Avec le paradigme NFV, les entreprises sont tentées de virtualiser et d'externaliser leurs infrastructures réseau vers le cloud afin de bénéficier des avantages de ce dernier. Pour les fournisseurs de services Cloud, la NFV ouvre la porte à de nouveaux modèles commerciaux très innovants et lucratifs tels que le Réseau en tant que service (NaaS) [LRMO16]. Un grand nombre de fournisseurs de cloud computing proposent déjà un large éventail de VNF prêts à l'emploi, par exemple l'équilibreur de charge [Gui18], Passerelle de transit [AWS] et Pare-feu réseau AWS. Les entreprises peuvent alors créer rapidement des services réseau efficaces en reliant divers VNF. Selon une étude menée par une recherche méticuleuse [Ltd20], le marché mondial des NFV devrait connaître une croissance annuelle de 34,9% pour atteindre 122 milliards d'ici 2027.

Malgré la reconnaissance unanime des avantages de l'externalisation des fonctions réseau vers le cloud par le biais de la NFV, la sécurité reste l'une des préoccupations essentielles et l'un des obstacles potentiels qui empêchent l'adoption à grande échelle de la NFV [PHS+18, ZAR22]. Le NFV élargit considérablement la surface d'attaque car il repose sur une large pile logicielle telle que des hyperviseurs (par exemple KVM, Docker), des outils d'automatisation et de gestion (par exemple OpenStack, Cloudify, Ansible) et des logiciels d'isolation du réseau et d'accélération du plan de données (par exemple OpenVswitch, FD.io). Chacun de ces logiciels peut exposer diverses vulnérabilités (par exemple CVE-2020-35498 et CVE-2020-3236), décuplant ainsi les possibilités pour un adversaire de violer les spécifications des services réseau de l'entreprise [ZAR22]. En outre, dans les plateformes NFV, les VNF appartenant à différents locataires (entreprises) partagent souvent le même serveur physique ou virtuel. Un locataire malveillant peut alors mener des attaques par canal latéral [AQK+19, AWJ20] contre les VNF des autres locataires pour voler ou corrompre des données sensibles, ou perturber les différentes fonctionnalités fournies par les VNF. Youngjoo Shin et al. [SKH20] ont montré que l'accès partagé au cache du processeur peut permettre à un locataire de déduire des informations telles que les règles de filtrage du pare-feu appliquées dans le service réseau d'un autre locataire.

Le manque de confiance des entreprises et des organisations dans les environnements NFV empêche l'externalisation des fonctions réseau vers le cloud. Cette lacune critique doit être comblée par le domaine de la recherche NFV, qui doit être alimenté par de nouvelles approches orientées sécurité pouvant être utilisées pour améliorer la posture de sécurité des environnements NFV existants.

B.2 Objectif et questions de la recherche

L'objectif de cette thèse est d'améliorer la sécurité des services NFV. Pour atteindre cet objectif, plusieurs questions de recherche doivent être répondues.

Tout d'abord, compte tenu du fait que les VNF sont des fonctions réseau logicielles fonctionnant sur des infrastructures virtualisées, les menaces de sécurité liées aux services réseau basés sur NFV peuvent être plus importantes que les services réseau traditionnels, allant des menaces de virtualisation génériques [AQK+19], des fonctions réseau physiques avant la virtualisation [BMMR+15], et des menaces introduites par la combinaison de la technologie de virtualisation avec la mise en réseau [AWJ20]

telles que l'orchestration et la violation des politiques. Le premier sous-objectif est donc d'établir une taxonomie complète des menaces spécifiques à la couche NFV en répondant aux questions de recherche suivantes :

- **QR-1**: Quelles sont les menaces pour la sécurité des infrastructures NFV?
- **QR-2**: Quelles couches de l'infrastructure NFV sont concernées par quelles menaces de sécurité?

Pour surmonter certaines des menaces et des vulnérabilités liées aux infrastructures NFV, plusieurs mécanismes de sécurité ont été proposés dans la littérature. L'un de ces mécanismes est le déploiement des politiques de contrôle d'accès. Par conséquent, comme deuxième sous-objectif, nous nous concentrons sur l'étude des mécanismes existants qui peuvent être utilisés pour appliquer des politiques de contrôle d'accès aux services NFV. Ensuite, afin d'évaluer et de comparer le déploiement des politiques de contrôle d'accès existantes dans les solutions de services NFV, le troisième sous-objectif consiste à identifier l'ensemble des propriétés pertinentes qui doivent être satisfaites pour améliorer efficacement la sécurité des services NFV. Nous remplissons les deux sous-objectifs précédents en répondant aux questions de recherche suivantes:

- **QR-3**: Quel est l'état de l'art en ce qui concerne le déploiement des politiques de contrôle d'accès sur les infrastructures de réseau virtualisées?
- **QR-4**: Quelles sont les propriétés pertinentes qui doivent être satisfaites par les mécanismes du contrôle d'accès pour renforcer la sécurité des services NFv dans les solutions existantes?

En répondant aux questions de recherche précédentes, nous observons que tous les mécanismes de contrôle d'accès existants sur les infrastructures virtualisées souffrent d'au moins une des limitations suivantes. Premièrement, aucune des approches existantes ne fournit de preuves formelles de l'exactitude du déploiement des politiques de contrôle d'accès. Deuxièmement, la plupart des approches existantes s'appuient sur des modèles de spécification qui ne sont pas assez expressifs pour traiter des politiques de haut niveau spécifiées à l'aide d'autres modèles de contrôle d'accès tels que RBAC [SCFY96], ORBAC [KBB+03], et ABAC [PFMP04]. Enfin, la plupart des approches existantes nécessitent la modification et la gestion de l'infrastructure NFV

(NFVI), ce qui les rend non conformes à l'architecture ETSI-NFV et donc difficilement utilisables en pratique. Par conséquent, pour surmonter la limitation précédente, nous nous concentrons sur la définition d'un mécanisme de déploiement du contrôle d'accès prouvé correct, hautement expressif et conforme à l'ETSI-NFV en étudiant les questions de recherche suivantes :

- **QR-5.1**: Comment définir un modèle de spécification de haut niveau suffisamment expressif qui permet d'exprimer correctement des exigences modélisées à l'aide de modèles de contrôle d'accès bien connus?

- **QR-5.2**: Comment raffiner correctement la politique de contrôle d'accès de haut niveau vers des exigences concrètes et déployables?

- **QR-5.3**: Comment définir un déploiement conforme à l'ETSI-NFV des exigences de niveau concret?

Les investigations que nous menons pour répondre aux questions de recherche précédentes nous permettent de constater que les politiques de contrôle d'accès de haut niveau sont souvent composées d'un ensemble d'autorisations et d'interdictions, ce qui peut donner lieu à des conflits et des exceptions dans la politique à appliquer. Par conséquent, comme cinquième sous-objectif, nous nous concentrons sur la définition d'une solution permettant un déploiement efficace des politiques de contrôle d'accès mixtes de haut niveau (c'est-à-dire des politiques qui contiennent à la fois des autorisations positives et négatives) contenant des exceptions dans une infrastructure virtualisée. Pour répondre au sous-objectif précédent, nous étudions les questions de recherche suivantes:

- **QR-5.4**: Les solutions proposées pour répondre aux QR-5.1, QR-5.2 et QR-5.3 peuvent-elles être utilisées pour déployer efficacement (c'est-à-dire en réduisant au mieux le nombre de règles dans la politique de niveau concret ainsi que le temps nécessaire pour évaluer une demande d'accès) des politiques de contrôle d'accès mixtes de haut niveau contenant des exceptions?

- **QR-5.5**: Comment pouvons-nous améliorer les solutions proposées pour traiter efficacement les politiques de contrôle d'accès mixtes de haut niveau contenant des exceptions?

Comme toute autre infrastructure virtualisée, les services NFV peuvent être compromis et partiellement contrôlés par un adversaire. La revue de la littérature dans la thèse montre que toutes les solutions existantes de déploiement des politiques de contrôle d'accès ne prennent en compte que les adversaires externes qui visent à contourner à distance la politique appliquée. Par conséquent, le sixième sous-objectif que nous considérons dans cette thèse se concentre sur la proposition d'une solution qui peut appliquer correctement les politiques de contrôle d'accès lorsque des adversaires externes et internes sont considérés. Pour répondre au sous-objectif précédent, nous étudions la question de recherche suivante:

- **QR-5.6**: Comment faire face à des adversaires internes et externes? Plus précisément, comment déployer correctement les politiques de contrôle d'accès lorsqu'une partie inconnue du service NFV cible est compromise et contrôlée par un adversaire?

Enfin, le déploiement de la politique de contrôle d'accès nécessite des ressources en termes de calcul et de stockage, et a souvent un impact sur la fonctionnalité fournie par le service NFV cible, c'est-à-dire en introduisant une latence due à l'analyse du trafic et au déploiement des règles. La revue de la littérature effectuée nous permet d'observer qu'aucune des solutions existantes de déploiement des politiques de contrôle d'accès sur les infrastructures virtualisées n'a envisagé de minimiser à la fois la consommation de ressources et l'impact lié au déploiement de la politique de contrôle d'accès. Par conséquent, le septième sous-objectif de cette thèse se concentre sur la proposition d'une solution de déploiement de la politique de contrôle d'accès qui permet de minimiser à la fois les ressources nécessaires aux points de déploiement de la politique et l'impact en termes de latence introduit par le déploiement de la politique de contrôle d'accès. Nous atteignons le sous-objectif précédent en étudiant la question de recherche suivante:

- **QR-5.7**: Comment définir un mécanisme de déploiement de la politique de contrôle d'accès qui minimise à la fois les ressources consommées et l'impact sur le service NFV cible causé par le déploiement de la politique de contrôle d'accès?

B.3 Méthodologies et contributions

Pour atteindre les objectifs de recherche susmentionnés et répondre aux questions de recherche définies ci-dessus, cette thèse apporte les contributions suivantes.

B.3.1 La sécurité des NFVs

Les technologies NFV sont de plus en plus utilisées par les opérateurs de communications électroniques pour répondre rapidement aux besoins du marché. Elles consistent à la virtualisation des fonctions réseau, qui sont le plus souvent des applications réseau de télécommunication, notamment des routeurs, des pare-feu, des équilibreurs de charge, etc. L'objectif principal est de permettre à ces fonctions de fonctionner comme des programmes logiciels ou des fonctions de réseau virtuelles (VNFs). Comme les fournisseurs de services ne présentent pas de structures internes, les utilisateurs peuvent avoir une capacité limitée de voir et de contrôler les ressources du réseau [LV94]. Ces limitations entraînent intrinsèquement d'importantes menaces pour la sécurité [DSVV17, LROT17, RLOH17, SECBC20, SECB$^+$20]. L'architecture NFV est composée de plusieurs composants (VNFs, NFVI, et NFV MANO). Chacun de ces composants peut introduire des menaces différentes.

Dans cette contribution, nous abordons l'ensemble des menaces liées aux différentes couches qui composent le NFVI. La figure B.1 donne une taxonomie des menaces existantes en fonction de la couche NFVI impactée.

Nous avons fais une comparaison des solutions existantes de déploiement des politiques de contrôle d'accès dans la littérature, cette étude nous a permis de faire les observations suivantes:

1. Toutes les approches existantes ne prennent en compte que les adversaires externes qui visent à contourner la politique de contrôle d'accès appliquée en interagissant à distance avec l'infrastructure cible. Aucune d'entre elles n'a étudié le déploiement des politiques de contrôle d'accès en présence d'un adversaire interne qui compromet une partie de l'infrastructure virtualisée.

2. Aucune des solutions proposées n'a fourni de preuves formelles de l'exactitude des méthodes de raffinement et de déploiement des politiques de contrôle d'accès utilisées.

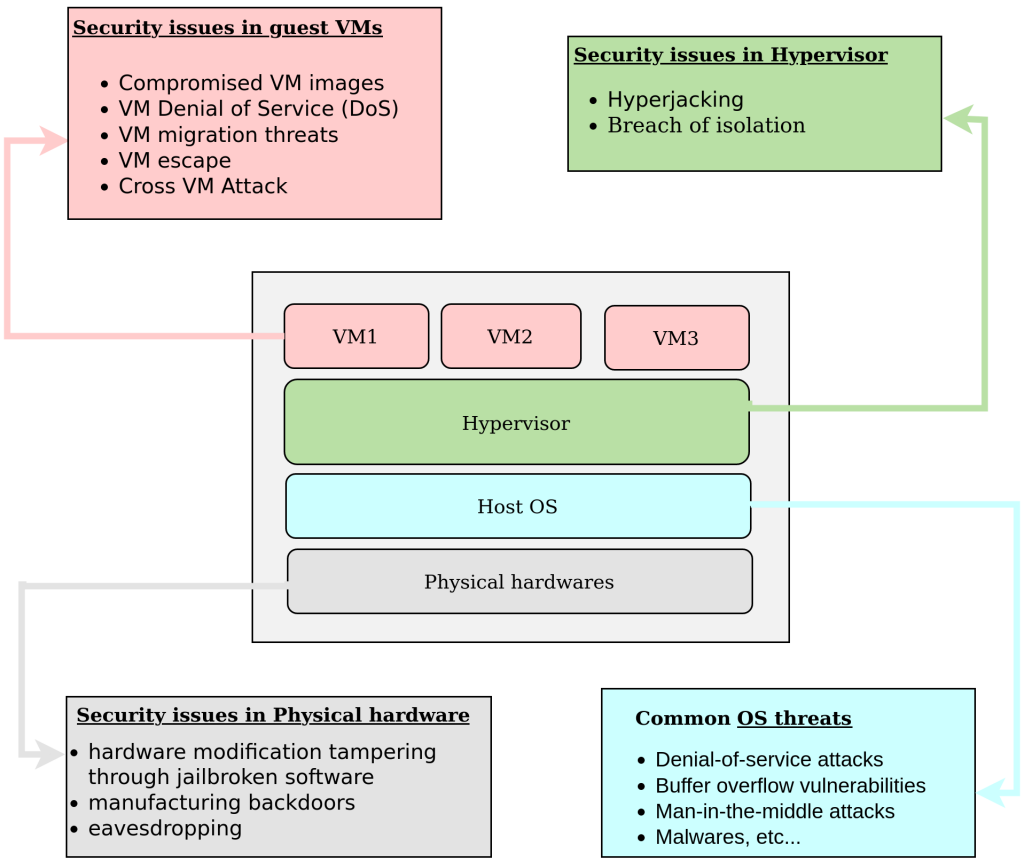


Figure B.1 – NFVI threats

3. Près de la moitié des solutions proposées ne fournissent aucun résultat théorique et/ou expérimental, ce qui rend impossible l'évaluation de leur efficacité. En particulier, seul un quart (5 sur 20) des solutions étudiées ont fourni des résultats démontrant leur efficacité.

4. Peu d'approches étudiées ont proposé des méthodes pour trouver la solution optimale de déploiement de la politique de contrôle d'accès. En outre, toutes les approches précédentes ont considéré soit l'optimisation des ressources (par exemple, les CPU, la mémoire) utilisées pour appliquer la politique, soit l'impact (par exemple, la latence introduite par les points de déploiement de la politique) sur l'infrastructure virtualisée en raison de déploiement de la politique. Aucune des solutions proposées n'a étudié à la fois l'optimisation des ressources nécessaires et les impacts introduits.

5. Environ la moitié des approches étudiées permettent une grande expressivité

de la politique de contrôle d'accès à appliquer tout en proposant une stratégie de déploiement légère (c'est-à-dire sans nécessiter la modification de l'infrastructure de virtualisation sur laquelle le service réseau virtualisé fonctionne). Cependant, seules deux des solutions proposées ont étudié la gestion des règles conflictuelles dans la politique à déployer.

6. Enfin, nous pouvons clairement observer qu'aucune des approches étudiées n'a été capable de remplir toutes les propriétés que nous avons identifiées.

B.3.2 Déploiement des politiques de contrôle d'accès sur les services NFV

Comme nous l'avons vu dans la première cotribution, plusieurs approches de déploiement de politiques de contrôle d'accès dans les infrastructures virtualisées ont été proposées. Nous avons constaté que la plupart de ces approches souffrent d'au moins une des limitations suivantes. Premièrement, aucune des approches existantes ne fournit de preuves formelles de l'exactitude du déploiement des politiques de contrôle d'accès. Deuxièmement, la plupart des approches existantes s'appuient sur des modèles de spécification qui ne sont pas assez expressifs pour traiter des politiques de haut niveau spécifiées à l'aide d'autres modèles de contrôle d'accès tels que RBAC, ABAC et ORBAC. Enfin, plusieurs approches existantes nécessitent la modification et la gestion du NFVI, ce qui les rend non conformes à l'architecture ETSI-NFV et donc difficilement utilisables en pratique. Par conséquent, pour surmonter les limitations précédentes, nous abordons dans cette contribution les questions de recherche suivantes :

- **QR-5.1:** Comment définir un modèle de spécification de haut niveau suffisamment expressif qui permet d'exprimer correctement les politiques spécifiées à l'aide d'autres modèles de contrôle d'accès bien connus tels que RBAC, ABAC et ORBAC?
- **QR-5.2:** Comment raffiner correctement la politique de contrôle d'accès de haut niveau vers des exigences déployables de niveau concret?
- **QR-5.3:** Comment définir un déploiement conforme à l'ETSI-NFV des exigences de niveau concret?

Pour répondre aux questions de recherche précédentes, nous proposons un modèle formel qui fournit un contrôle d'accès défini par logiciel en tant que capacité de service pour les services de réseau virtualisés. Tout d'abord, nous définissons un modèle de spécification expressif permettant d'exprimer les exigences de contrôle d'accès de haut niveau à appliquer aux services de réseaux virtualisés. Nous montrons ensuite que notre modèle de spécification peut exprimer correctement les exigences de contrôle d'accès spécifiées à l'aide d'autres modèles de contrôle d'accès tels que RBAC, ABAC et ORBAC. Ensuite, nous proposons une méthode prouvée correcte pour raffiner les exigences de contrôle d'accès de haut niveau vers une spécification concrète d'application de type et domaine (DTE). Enfin, notre modèle définit une méthode de déploiement efficace et conforme à la norme ETSI-NFV, comme l'illustrent les différentes évaluations expérimentales. En outre, notre méthode de déploiement de la politique est évolutive car il est possible d'ajouter autant de points de déploiement que nécessaire (par exemple, à des fins d'équilibrage de charge) sans affecter le fonctionnement des services de réseau.

B.3.3 Un DTE basé sur les priorités pour la gestion des exceptions

Comme nous l'avons vu dans la deuxième contribution, plusieurs modèles sont proposés pour gérer l'accès dans et entre les services NFV. Malheureusement, les techniques identifiées souffrent d'au moins une des limitations suivantes : (1) La manière dont la transformation de la politique de haut niveau vers une politique concrète déployable est effectuée n'est pas claire (par exemple, [Jae15, TCCM17]). (2) Le manque de généralité, soit en exigeant la modification de l'infrastructure NFV pour permettre le déploiement de la politique de contrôle d'accès dans les services NFV (par exemple, [PHMZ16, PTH+17]), soit en ne prenant en charge qu'un modèle de politique de contrôle d'accès spécifique (par exemple, [PHMZ16]).

Pour surmonter les limitations précédentes, nous proposons un modèle formel basé sur Domain Type Enforcement (DTE), permettant de gérer le déploiement de politiques exprimées à l'aide des modèles de contrôle d'accès les plus connus tels que RBAC, ORBAC et ABAC. Ce modèle fournit une méthode formelle et efficace pour déployer des politiques de contrôle d'accès au sein des services NFV sans nécessiter la modification de l'infrastructure NFV. Cependant, le modèle proposé précédemment a été

principalement conçu pour déployer des politiques de contrôle d'accès fermées. Ces dernières sont connues pour souffrir d'un manque d'expressivité, c'est-à-dire lorsqu'il faut exclure des cas spécifiques (par exemple, des exceptions) de règles générales qui devraient toujours s'appliquer [ACCB07, CCB18]. À titre d'illustration, considérons un service NFV composé de n VNFs (VNF$_1$, $\cdots$, VNF$_n$) et supposons que tous les VNFs, à l'exception de VNF$_i$, puissent communiquer avec un service spécifique *s*. En utilisant une politique mixte, les exigences précédentes peuvent être exprimées à l'aide des deux règles suivantes :

- VNF$_i$ n'est pas autorisé à accéder à *s*.
- Tous les VNFs sont autorisés à accéder à *s*.

Cependant, l'utilisation de politiques fermées pour exprimer les exigences de contrôle d'accès précédentes conduira à une politique composée de $n-1$ règles d'autorisation de haut niveau, qui peuvent introduire une latence élevée lorsqu'elles sont déployées au service NFV.

Néanmoins, bien que les politiques mixtes soient bien adaptées pour exprimer des exigences d'accès de haut niveau contenant des exceptions, il a été démontré dans [ACCB07] que les politiques mixtes de contrôle d'accès de haut niveau contenant des exceptions conduisent souvent à des configurations concrètes assez complexes. Par conséquent, la question de recherche à laquelle nous voulons répondre dans cette partie est la suivante :

- **QR-5.4:** Pouvons-nous utiliser la solution de déploiement des politiques de contrôle d'accès proposée pour déployer efficacement des politiques de contrôle d'accès mixtes et complexes de haut niveau contenant des exceptions?

Ensuite, si la réponse à la question précédente est négative, la deuxième question de recherche à laquelle il faut répondre est la suivante :

- **QR-5.5:** Comment pouvons-nous améliorer la solution proposée pour traiter efficacement les politiques de contrôle d'accès mixtes et complexes de haut niveau contenant des exceptions ?

Pour répondre aux questions de recherche QR-5.4 et QR-5.5, nous étudions la gestion des exceptions de contrôle d'accès dans les spécifications concrètes basées sur

DTE. Ensuite, nous proposons une solution pour permettre un déploiement propre et efficace de politiques de contrôle d'accès complexes contenant des exceptions et/ou des règles de conflit sur les services NFV. Notre modèle permet un déploiement propre dans le sens où, par rapport à la politique de sécurité de haut niveau exprimée à l'aide du modèle de spécification proposé à déployer, il n'introduit pas de règles de bas niveau supplémentaires, ce qui permet à l'administrateur de sécurité de comprendre et de mettre à jour directement la politique de niveau concret déployée. Notre modèle s'appuie sur une approche prouvée correcte pour la gestion des exceptions dans la spécification DTE. Les évaluations empiriques réalisées montrent que le modèle DTE basé sur les priorités proposé est plus efficace pour déployer des politiques importantes et complexes qui contiennent des exceptions.

B.3.4 Déploiement optimal du contrôle d'accès sur les services NFV

Comme nous l'avons vu dans la troisième contribution, la sécurité est une préoccupation majeure dans les réseaux NFV car ils sont sensibles aux menaces liées aux vulnérabilités du réseau, de la virtualisation et des utilisateurs malveillants. Une fois exploitées, ces vulnérabilités conduisent souvent à une utilisation abusive affectant les capacités fournies par les services NFV ainsi que la sécurité de leurs utilisateurs finaux.

Dans les précédentes contributions de cette thèse, nous proposons un cadre de déploiement du contrôle d'accès sur les services NFV pour atténuer les menaces soulevées principalement par les vulnérabilités liées aux utilisateurs malveillants (par exemple, accès/privilège non autorisé, flux non autorisé entre VNFs). Les approches proposées ont pris en compte divers critères liés à la spécification, au raffinement et au déploiement des politiques de contrôle d'accès, tels que l'expressivité du modèle de contrôle d'accès proposé, la modélisation formelle et la vérification de l'exactitude du raffinement et du déploiement des politiques de contrôle d'accès. Cependant, la plupart des solutions susmentionnées n'ont pas pris en compte deux critères fondamentaux.

Tout d'abord, les services NFV, comme toute autre infrastructure virtualisée, peuvent être compromis et partiellement contrôlés par un adversaire qui contrôle totalement le service NFV cible, c'est-à-dire que tous les VNF qui composent le service NFV

sont contrôlés par l'adversaire, alors il n'y a rien à protéger par une approche de contrôle d'accès. Cependant, les solutions existantes ainsi que celles que nous avons proposées dans les précédentes contributions ne prennent en compte que les adversaires externes qui visent à contourner à distance la politique déployée. Par conséquent, la capacité de gérer les adversaires internes et externes est un critère important à prendre en compte lors de la définition de solutions pour le déploiement de la politique de contrôle d'accès sur les services NFV. Deuxièmement, le déploiement de la politique de contrôle d'accès nécessite des ressources en termes de calcul et de stockage, et a souvent un impact sur la fonctionnalité fournie par le service NFV cible, par exemple en introduisant une latence due à l'analyse du trafic et au déploiement des règles. Néanmoins, toutes les solutions existantes de déploiement de la politique de contrôle d'accès sur l'infrastructure NFV, y compris celles que nous avons proposées dans la deuxième et la quatrième contribution, ne prennent pas en compte la minimisation de la consommation de ressources et de l'impact lié au déploiement de la politique de contrôle d'accès, ce qui nous semble être un critère important à prendre en compte lors de la conception de solutions pour le déploiement de la politique de contrôle d'accès sur les services NFV.

A la lumière des observations précédentes, les questions de recherche auxquelles nous cherchons à répondre dans cette contribution sont les suivantes:

- **QR-5.6:** Comment améliorer les solutions proposées précédemment pour faire face aux adversaires internes? Plus précisement, comment déployer correctement les politiques de contrôle d'accès lorsqu'une partie inconnue du service NFV est contrôlée par un adversaire?
- **QR-5.7:** Comment étendre notre cadre de déploiement de la politique de contrôle d'accès pour minimiser à la fois les ressources consommées et l'impact causé par le déploiement de la politique de contrôle d'accès?

Pour répondre aux questions de recherche précédentes, nous proposons les contributions suivantes :

- Nous considérons un modèle d'adversaire fort : Contrairement aux approches proposées qui ne considèrent qu'un adversaire externe, nous supposons que nous avons affaire à un adversaire interne qui peut contrôler un ou plusieurs nœuds (VNF) qui composent le service NFV. En outre, nous supposons que les nœuds compromis ne sont pas connus.

- Nous proposons une modélisation formelle du déploiement optimal de la politique de contrôle d'accès permettant de modéliser et de quantifier les ressources consommées ainsi que l'impact en termes de latence qui doit être généré par le déploiement de la politique de contrôle d'accès. Le modèle proposé permet de prouver formellement l'exactitude du déploiement de la politique de contrôle d'accès.
- Nous montrons que le problème du déploiement optimal des politiques de contrôle d'accès est un problème d'optimisation multi-objectifs non linéaire et nous utilisons une amélioration de l'algorithme génétique de tri non dominé NSGA II [WSZ+18] pour le résoudre.
- Nous menons une expérience dans un environnement Internet émulé, NS-3 [ns3], pour évaluer les solutions de déploiement de la politique de contrôle d'accès.

BIBLIOGRAPHY

[AA19] Ibraheem Alothaimeen and David Arditi. Overview of multi-objective optimization approaches in construction project management. *Multicriteria Optimization-Pareto-Optimality and Threshold-Optimality*, 2019.

[ABET16] Omnia AbdElRahem, Ayman M Bahaa-Eldin, and Ayman Taha. Virtualization security: A survey. In *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, pages 32–40. IEEE, 2016.

[ACCB07] JG Alfaro, Frederic Cuppens, and Nora Cuppens-Boulahia. Management of exceptions on access control policies. In *IFIP International Information Security Conference*, pages 97–108. Springer, 2007.

[AEKEBB+03] Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miege, Claire Saurel, and Gilles Trouessin. Or-bac: un modèle de contrôle d'accès basé sur les organisations. *Cahiers francophones de la recherche en sécurité de l'information*, 1:30–43, 2003.

[AH21] Yunusa Simpa Abdulsalam and Mustapha Hedabou. Security and privacy in cloud computing: Technical review. *Future Internet*, 14(1):11, 2021.

[AKS13] Naveed Ahmad, Ayesha Kanwal, and Muhammad Awais Shibli. Survey on secure live virtual machine (vm) migration in cloud. In *2013 2nd National Conference on Information Assurance (NCIA)*, pages 101–106. IEEE, 2013.

[All90] Arnold O Allen. *Probability, statistics, and queueing theory*. Gulf Professional Publishing, 1990.

[AQK+19] Ahmed Osama Fathy Atya, Zhiyun Qian, Srikanth V Krishnamurthy, Thomas La Porta, Patrick McDaniel, and Lisa M Marvel. Catch me if you

can: A closer look at malicious co-residency on the cloud. *IEEE/ACM Transactions on Networking*, 27(2):560–576, 2019.

[ASM19] Rashid Amin, Nadir Shah, and Waqar Mehmood. Enforcing optimal acl policies using k-partite graph in hybrid sdn. *Electronics*, 8(6):604, 2019.

[ASS13] MR Anala, Jyoti Shetty, and G Shobha. A framework for secure live migration of virtual machines. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 243–248. IEEE, 2013.

[AWJ20] Nawaf Alhebaishi, Lingyu Wang, and Sushil Jajodia. Modeling and mitigating security threats in network functions virtualization (nfv). In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 3–23. Springer, 2020.

[AWS] Aws transit gateway. `https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html`. Accessed: Juin 14,2022.

[AZZ+20] Ihsan H Abdulqadder, Shijie Zhou, Deqing Zou, Israa T Aziz, and Syed Muhammad Abrar Akber. Multi-layered intrusion detection and prevention in the sdn/nfv enabled cloud of 5g networks using ai-based defense mechanisms. *Computer Networks*, 179:107364, 2020.

[BBKL14] Tobias Binz, Uwe Breitenbücher, Oliver Kopp, and Frank Leymann. Tosca: portable automated deployment and management of cloud applications. In *Advanced Web Services*, pages 527–549. Springer, 2014.

[BBS+12] Carel Nicolaas Bezuidenhout, Shamim Bodhanya, Thawani Sanjika, Milindi Sibomana, and Gordon Louis Nelson Boote. Network-analysis approaches to deal with causal complexity in a supply network. *International Journal of Production Research*, 50(7):1840–1849, 2012.

[BD01] H-G Beyer and Kalyanmoy Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 5(3):250–270, 2001.

[BHNGN+19] Antonio Benítez-Hidalgo, Antonio J. Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51, dec 2019.

[BLP+15] Cataldo Basile, Antonio Lioy, Christian Pitscheider, Fulvio Valenza, and Marco Vallini. A novel approach for integrating security policy enforcement with dynamic network virtualization. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–5. IEEE, 2015.

[BMMR+15] A Belmonte Martin, L Marinos, E Rekleitis, G Spanoudakis, and NE Petroulakis. Threat landscape and good practice guide for software defined networks/5g. 2015.

[BMS+20] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. Automated optimal firewall orchestration and configuration in virtualized networks. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2020.

[BMSV20] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. Short paper: Automatic configuration for an optimal channel protection in virtualized networks. In *Proceedings of the 2nd Workshop on Cyber-Security Arms Race*, pages 25–30, 2020.

[BMSV21] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. A novel approach for security function graph configuration and deployment. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 457–463. IEEE, 2021.

[BP+03] Michael R Bussieck, Armin Pruessner, et al. Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News*, 14(1):19–22, 2003.

[BR+11] Anthony Bisong, M Rahman, et al. An overview of the security concerns in enterprise cloud computing. *arXiv preprint arXiv:1101.5613*, 2011.

[BSS+96] Lee Badger, Daniel F Sterne, David L Sherman, Kenneth M Walker, and Sheila A Haghighat. A domain and type enforcement unix prototype. *Computing Systems*, 9(1):47–83, 1996.

[BVL+19] Cataldo Basile, Fulvio Valenza, Antonio Lioy, Diego R Lopez, and Antonio Pastor Perales. Adding support for automatic enforcement of security policies in nfv networks. *IEEE/ACM Transactions on Networking*, 27(2):707–720, 2019.

[CCB18] Frédéric Cuppens and Nora Cuppens-Boulahia. Stratification based model for security policy with exceptions and contraries to duty. In *From Database to Cyber Security*, pages 78–103. Springer, 2018.

[CGS+17] Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K Awasthi, Emmanuel S Pilli, and Divya Kapil. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing*, 6(1):1–41, 2017.

[Cho13] Te-Shun Chou. Security threats on cloud computing vulnerabilities. *AIRCC's International Journal of Computer Science and Information Technology*, 5(3):79–88, 2013.

[Cla00] Paul C Clark. Policy-enhanced linux. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2000.

[CSS+12] Ying Chen, Qingni Shen, Pengfei Sun, Yangwei Li, Zhong Chen, and Sihan Qing. Reliable migration module in trusted cloud based on security level-design and implementation. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2230–2236. IEEE, 2012.

[DB05] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[DD90] Sajal K Das and Narsingh Deo. Parallel hungarian algorithm. *Computer Systems Science and Engineering*, 5(3), 1990.

[DD98] Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria

optimization problems. *SIAM journal on optimization*, 8(3):631–657, 1998.

[DKP+17] Sevil Dräxler, Holger Karl, Manuel Peuster, Hadi Razzaghi Kouchaksaraei, Michael Bredel, Johannes Lessmann, Thomas Soenen, Wouter Tavernier, Sharon Mendel-Brin, and George Xilouris. Sonata: Service programming and orchestration for virtualized software networks. In *2017 IEEE international conference on communications workshops (ICC Workshops)*, pages 973–978. IEEE, 2017.

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[DSPA18] Prachi Deshpande, Subhash Chander Sharma, Sateesh K Peddoju, and Ajith Abraham. Security and service assurance issues in cloud environment. *International Journal of System Assurance Engineering and Management*, 9(1):194–207, 2018.

[DSVV17] Luca Durante, Lucia Seno, Fulvio Valenza, and Adriano Valenzano. A model for the analysis of security policies in service function chains. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6. IEEE, 2017.

[DTM10] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel. Infrastructure as a service security: Challenges and solutions. In *2010 the 7th International Conference on Informatics and Systems (INFOS)*, pages 1–8. IEEE, 2010.

[E+14] NFVISG ETSI et al. Network functions virtualisation (nfv); management and orchestration. *NFV-MAN*, 1:v0, 2014.

[Ehr05] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[ELP00] MM Eusuff, KE Lansey, and F Pasha. Shuffled frog leaping algorithm: a memetic meta-heuristic for combinatorial optimization. *J. Heuristics*, 2000.

[EMEBHM16] Khalid El Makkaoui, Abdellah Ezzati, Abderrahim Beni-Hssane, and Cina Motamed. Cloud security and privacy model for providing secure cloud services. In *2016 2nd international conference on cloud computing technologies and applications (CloudTech)*, pages 81–86. IEEE, 2016.

[ENI22] ENISA. Nfv security in 5g - challenges and best practices. https://www.enisa.europa.eu/publications/nfv-security-in-5g-challenges-and-best-practices/@@download/fullReport, 2022. Accessed: Juin 14,2022.

[Ers13] Mehmet Ersue. Etsi nfv management and orchestration-an overview. *Presentation at the IETF*, 88, 2013.

[FJKK17] Mahdi Daghmehchi Firoozjaei, Jaehoon Paul Jeong, Hoon Ko, and Hyoungshick Kim. Security challenges with network functions virtualization. *Future Generation Computer Systems*, 67:315–324, 2017.

[FSG+01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[G+16] Cybersecurity Working Group et al. White paper: Considerations for securing sdn/nfv, 2016.

[GMW10] Oscar Garcia-Morchon and Klaus Wehrle. Modular context-aware access control for medical sensor networks. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pages 129–138, 2010.

[GS18] Daniel Guija and Muhammad Shuaib Siddiqui. Identity and access control for micro-services based 5g nfv platforms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.

[Gui18] Developer Guide. Amazon elastic load balancing. 2018.

[Gun18] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

[GWS10] Bernd Grobauer, Tobias Walloschek, and Elmar Stocker. Understanding cloud computing vulnerabilities. *IEEE Security & privacy*, 9(2):50–57, 2010.

[GXWS20] Xing Gao, Jidong Xiao, Haining Wang, and Angelos Stavrou. Understanding the security implication of aborting live migration. *IEEE Transactions on Cloud Computing*, 2020.

[Har12] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[HCSL12] Yu-Lun Huang, Borting Chen, Ming-Wei Shih, and Chien-Yu Lai. Security impacts of virtualization on a network testbed. In *2012 IEEE Sixth International Conference on Software Security and Reliability*, pages 71–77. IEEE, 2012.

[HGJL15] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE communications magazine*, 53(2):90–97, 2015.

[HKKT10] Kevin Hamlen, Murat Kantarcioglu, Latifur Khan, and Bhavani Thuraisingham. Security issues for cloud computing. *International Journal of Information Security and Privacy (IJISP)*, 4(2):36–48, 2010.

[HNG94] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

[HRFMF13] Keiko Hashizume, David G Rosado, Eduardo Fernández-Medina, and Eduardo B Fernandez. An analysis of security issues for cloud computing. *Journal of internet services and applications*, 4(1):1–13, 2013.

[IMZ16] Tariqul Islam, D Manivannan, and Sherali Zeadally. A classification and characterization of security threats in cloud computing. *International Journal of Next-Generation Computing*, pages 01–17, 2016.

[Jae15] Bernd 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. IEEE, 2015.

[JJW+21] Wei Jiang, Wanchun Jiang, Jianxin Wang, Jianliang Gao, and Neal Xiong. Orp: An online rule placement scheme to optimize the traffic overhead for data center networks. *IEEE Transactions on Network Science and Engineering*, 8(3):2183–2197, 2021.

[JSGI09] Meiko Jensen, Jörg Schwenk, Nils Gruschka, and Luigi Lo Iacono. On technical security issues in cloud computing. In *2009 IEEE international conference on cloud computing*, pages 109–116. Ieee, 2009.

[Kak04] MM Raghuwanshi1and OG Kakde. Survey on multiobjective evolutionary and real coded genetic algorithms. In *Proceedings of the 8th Asia Pacific symposium on intelligent and evolutionary systems*, pages 150–161. Citeseer, 2004.

[KBB+03] Anas Abou El Kalam, R El Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miege, Claire Saurel, and Gilles Trouessin. Organization based access control. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 120–131. IEEE, 2003.

[KE95] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[Ken53] David G Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953.

[KSK14] ES Phalguna Krishna, E Sandhya, and M Ganesh Karthik. Managing ddos attacks on virtual machines by segregated policy management. *Global Journal of Computer Science and Technology*, 2014.

[KYNL20] Sunghwan Kim, Seunghyun Yoon, Jargalsaikhan Narantuya, and Hyuk Lim. Secure collecting, optimizing, and deploying of firewall rules in software-defined networks. *IEEE Access*, 8:15166–15177, 2020.

[LDB+21] Rui Li, Bertrand Decocq, Anne Barros, Yiping Fang, and Zhiguo Zeng. Complexity in 5g network applications and use cases. In *31st European Safety and Reliability Conference*, pages 3054–3061. Research Publishing Services, 2021.

[LL21] Pei-Hsuan Lee and Fuchun Joseph Lin. Tackling iot scalability with 5g nfv-enabled network slicing. *Advances in Internet of Things*, 11(3):123–139, 2021.

[LPS13] Seungjoon Lee, Manish Purohit, and Barna Saha. Firewall placement in cloud data centers. In *Proceedings of the 4th annual Symposium on Cloud Computing*, pages 1–2, 2013.

[LQ16] Xin Li and Chen Qian. An nfv orchestration framework for interference-free policy enforcement. In *2016 IEEE 36th international conference on distributed computing systems (ICDCS)*, pages 649–658. IEEE, 2016.

[LRMO16] Diego Lopez, Andy Reid, Antonio Manzalini, and Marie-Paule Odini. Impact of sdn/nfv on business models. *IEEE Softwarization*, 2016.

[LROT17] Shankar Lal, Sowmya Ravidas, Ian Oliver, and Tarik Taleb. Assuring virtual network function image integrity and host sealing in telco cloue. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[LTD17] Shankar Lal, Tarik Taleb, and Ashutosh Dutta. Nfv: Security threats and best practices. *IEEE Communications Magazine*, 55(8):211–217, 2017.

[Ltd20] Meticulous Market Research Pvt. Ltd. Network function virtualization (nfv) market by component, virtualized network function, application, end user - global forecast to 2027. Technical report, Meticulous Research, 2020.

[LV94] Thomas DC Little and Dinesh Venkatesh. Prospects for interactive video-on-demand. *IEEE multimedia*, 1(3):14–24, 1994.

[LWZY06] Bo Long, Xiaoyun Wu, Zhongfei Zhang, and Philip S Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the 12th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 317–326, 2006.

[MA17] Chirag N Modi and Kamatchi Acha. Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review. *the Journal of Supercomputing*, 73(3):1192–1234, 2017.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74, 2008.

[MGT+15] Jon Matias, Jokin Garay, Nerea Toledo, Juanjo Unzilla, and Eduardo Jacob. Toward an sdn-enabled nfv architecture. *IEEE Communications Magazine*, 53(4):187–193, 2015.

[MI+95] Tadahiko Murata, Hisao Ishibuchi, et al. Moga: multi-objective genetic algorithms. In *IEEE international conference on evolutionary computation*, volume 1, pages 289–294. IEEE Piscataway, NJ, USA, 1995.

[MR20] Andrés F Murillo and Sandra Rueda. Access control policies for network function virtualization environments in industrial control systems. In *2020 4th Conference on Cloud and Internet of Things (CIoT)*, pages 17–24. IEEE, 2020.

[MRD18] Leopoldo AF Mauricio, Marcelo G Rubinstein, and Otto Carlos MB Duarte. Aclflow: An nfv/sdn security framework for provisioning and managing access control lists. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 44–51. IEEE, 2018.

[MS14] Sunilkumar S Manvi and Gopal Krishna Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of network and computer applications*, 41:424–440, 2014.

[ns3] ns-3 | a discrete-event network simulator for internet systems.

[odl] The opendaylight platform. `https://www.opendaylight.org/`. Accessed: January 30th,2019.

[ofm] Openflow manager. https://github.com/CiscoDevNet/OpenDaylight-Openflow-App. Accessed: January 30th,2019.

[OKJ+17] Sanghak Oh, Eunsoo Kim, Jaehoon Jeong, Hoon Ko, and Hyoungshick Kim. A flexible architecture for orchestrating network security functions to support high-level security policies. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, pages 1–5, 2017.

[Pag13] Pierluigi Paganini. Hardware attacks, backdoors, and electronic component qualification. *InfoSec Institute*, 2013.

[Pat19] Montida Pattaranantakul. *Moving towards software-defined security in the era of NFV and SDN*. PhD thesis, Université Paris-Saclay, 2019.

[PBB13] Gábor Pék, Levente Buttyán, and Boldizsár Bencsáth. A survey of security issues in hardware virtualization. *ACM Computing Surveys (CSUR)*, 45(3):1–34, 2013.

[PFMP04] Torsten Priebe, Eduardo B Fernández, Jens I Mehlau, and Günther Pernul. A pattern system for access control. In *Research Directions in Data and Applications Security XVIII*, pages 235–249. Springer, 2004.

[PHMZ16] Montida Pattaranantakul, Ruan He, Ahmed Meddahi, and Zonghua Zhang. Secmano: Towards network functions virtualization (nfv) based security management and orchestration. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 598–605. IEEE, 2016.

[PHS+18] Montida Pattaranantakul, Ruan He, Qipeng Song, Zonghua Zhang, and Ahmed Meddahi. Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys & Tutorials*, 20(4):3330–3368, 2018.

[PHZ+18] 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*, 2018.

[PTH+17] Montida Pattaranantakul, Yuchia Tseng, Ruan He, Zonghua Zhang, and Ahmed Meddahi. A first step towards security extension for nfv orchestrator. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 25–30, 2017.

[RAB+16] François Reynaud, François-Xavier Aguessy, Olivier Bettan, Mathieu Bouet, and Vania Conan. Attacks against network functions virtualization and software-defined networking: State-of-the-art. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 471–476. IEEE, 2016.

[RLOH17] Sowmya Ravidas, Shankar Lal, Ian Oliver, and Leo Hippelainen. Incorporating trust in nfv: Addressing the challenges. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 87–91. IEEE, 2017.

[RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212, 2009.

[SAD+12] Ivan Studnia, Eric Alata, Yves Deswarte, Mohamed Kaâniche, and Vincent Nicomette. Survey of security problems in cloud computing virtual machines. In *Computer and Electronics Security Applications Rendezvous (C&ESAR 2012). Cloud and security: threat or opportunity*, pages p–61, 2012.

[SAS12] Jyoti Shetty, MR Anala, and G Shobha. A survey on techniques of secure live migration of virtual machine. *International Journal of Computer Applications*, 39(12):34–39, 2012.

[SBJ+] Nat Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. OpenID connect core 1.0. `https://openid.net/specs/openid-connect-core-1_0.html`.

[SCFY96] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[SD94] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[SECB+20] Manel Smine, David Espes, Nora Cuppens-Boulahia, Frédéric Cuppens, and Marc-Oliver Pahl. A priority-based domain type enforcement for exception management. In *International Symposium on Foundations and Practice of Security*, pages 65–81. Springer, 2020.

[SECBC20] Manel Smine, David Espes, Nora Cuppens-Boulahia, and Frédéric Cuppens. Network functions virtualization access control as a service. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 100–117. Springer, 2020.

[SECCB20] Luis Suarez, David Espes, Frédéric Cuppens, and Nora Cuppens-Boulahia. Formalization of a security access control model for the 5g system. 2020.

[SJK+14] Nick Shelly, Ethan J Jackson, Teemu Koponen, Nick McKeown, and Jarno Rajahalme. Flow caching for high entropy packet fields. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 151–156, 2014.

[SKH20] Youngjoo Shin, Dongyoung Koo, and Junbeom Hur. Inferring firewall rules by cache side-channel analysis in network function virtualization. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1798–1807. IEEE, 2020.

[SLCL09] Monirul I Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. Secure in-vm monitoring using hardware virtualization. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 477–487, 2009.

[SM14] Norshazrul Azman Bin Sulaiman and Hideo Masuda. Evaluation of a secure live migration of virtual machines using ipsec implementation. In *2014 IIAI 3rd International Conference on Advanced Applied Informatics*, pages 687–693. IEEE, 2014.

[Smi] Manel Smine. DTE engine. `https://github.com/msmine/vnf-access-control-as-a-service`.

[SP97] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[SV13] Thomas L Saaty and Luis G Vargas. The analytic network process. In *Decision making with the analytic network process*, pages 1–40. Springer, 2013.

[Tac17] O Tacker. Tacker-openstack nfv orchestration, 2017. Accessed: January 25th,2019.

[TCCM17] Tran Quang Thanh, Stefan Covaci, Marius Corici, and Thomas Magedanz. Access control management and orchestration in nfv environment. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–2. IEEE, 2017.

[TKJ16] Tarik Taleb, Adlen Ksentini, and Riku Jantti. " anything as a service" for 5g mobile systems. *IEEE Network*, 30(6):84–91, 2016.

[VLA87] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[VNF] Vnffg. `https://docs.openstack.org/tacker/latest/user/vnffg_usage_guide.html`. Accessed: January 1,2019.

[WBO15] Candid Wueest, Mario Ballano Barcena, and Laura O'Brien. Mistakes in the iaas cloud could put your data at risk. *Symantec*, 2015.

[WCHW08] Chun Wang, Hao-zhong Cheng, Ze-chun Hu, and Yi Wang. Distribution system optimization planning based on plant growth simulation algorithm. *Journal of Shanghai Jiaotong University (Science)*, 13(4):462–467, 2008.

[WSZ+18] Yanfeng Wang, Yongpeng Shen, Xuncai Zhang, Guangzhao Cui, and Junwei Sun. An improved non-dominated sorting genetic algorithm-ii (insga-ii) applied to the design of dna codewords. *Mathematics and Computers in Simulation*, 151:131–139, 2018.

[XX12] Zhifeng Xiao and Yang Xiao. Security and privacy in cloud computing. *IEEE communications surveys & tutorials*, 15(2):843–859, 2012.

[YF16] Wei Yang and Carol Fung. A survey on security in network functions virtualization. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 15–19. IEEE, 2016.

[YG15] Sadiq T Yakasai and Chris G Guy. Flowidentity: Software-defined network access control. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 115–120. IEEE, 2015.

[YMH+18] Rehana Yasmin, Mohammad Reza Memarian, Shohreh Hosseinzadeh, Mauro Conti, and Ville Leppänen. Investigating the possibility of data leakage in time of live vm migration. In *Cyber Threat Intelligence*, pages 259–279. Springer, 2018.

[YT05] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.

[ZAR22] Moubarak Zoure, Toufik Ahmed, and Laurent Réveillére. Network services anomalies in nfv: Survey, taxonomy, and verification methods. *IEEE Transactions on Network and Service Management*, 2022.

[ZCP+15] Letterio Zuccaro, Federico Cimorelli, F Delli Priscoli, C Gori Giorgi, Salvatore Monaco, and Vincenzo Suraci. Distributed control in virtualized networks. *Procedia Computer Science*, 56:276–283, 2015.

[ZLJ21] Wenhui Zhang, Peng Liu, and Trent Jaeger. Analyzing the overhead of file protection by linux security modules. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 393–406, 2021.

[ZPL+20] Danyang Zheng, Chengzong Peng, Xueting Liao, Ling Tian, Guangchun Luo, and Xiaojun Cao. Towards latency optimization in hybrid service function chain composition and embedding. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1539–1548. IEEE, 2020.

[ZT98] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report*, 43, 1998.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : La sécurité définie par le logiciel pour la virtualisation des fonctions réseaux

Mot clés : Virtualisation des fonctions réseau (NFV), contrôle d'accès (AC), déploiement des politiques, gestion des exceptions, optimisation

Résumé : La technologie NFV (Network Function Virtualization) a été proposé pour améliorer la flexibilité du déploiement et la rentabilité des services de réseau et elle permet un meilleur partage des informations entre leurs composants. Malgré les avantages susmentionnés, les infrastructures NFV existantes souffrent de plusieurs problèmes de sécurité. L'objectif de cette thèse est d'améliorer la sécurité des services NFVs. Pour atteindre cet objectif, nous avons proposé un certain nombre de contributions. Tout d'abord, nous avons étudié les problèmes et les menaces de sécurité existants et potentiels dans les infrastructures NFVs. Ensuite, nous les avons classés en fonction des composants qui sont affectés par ces menaces. Puis, nous avons étudié les différents mécanismes de sécurité qui peuvent être utilisés pour réduire ces risques. Cette étude nous a permis de réaliser que le déploiement des politiques de contrôle d'accès au niveau des services NFV permet d'atténuer plusieurs problèmes de sécurité. Ce constat nous a conduit à examiner les solutions existantes pour le déploiement des politiques de contrôle d'accès dans les infrastructures NFV. Troisièmement, nous fournissons un aperçu comparatif complet des solutions existantes en se basant sur plusieurs propriétés, y compris le modèle d'adversaire considéré, l'efficacité, la véracité et l'optimalité

Title: Software-defined Security for Network Function Virtualization

Keywords: Network Functions Virtualization (NFV), Access Control (AC), policy enforcement (PE), Exception management, Optimization

Abstract: Network Function Virtualization (NFV) technology has been proposed to improve the deployment flexibility and cost effectiveness of network services and allows for better information sharing between their components. Despite the aforementioned benefits, existing NFV infrastructures suffer from several security issues. The objective of this thesis is to improve the security of NFV services. To achieve this goal, we proposed a number of contributions. First, we studied the existing and potential security problems and threats in NFV infrastructures. Then, we classified them according to the components that are affected by these threats. Then, we studied the different security mechanisms that can be used to reduce these risks. This study allowed us to realize that the deployment of access control policies at the NFV service level can mitigate several security issues. This realization led us to examine existing solutions for deploying access control policies in NFV infrastructures. Third, we provide a comprehensive comparative overview of existing solutions based on several properties, including the adversary model considered, effectiveness, veracity, and optimality.