Towards a Content-oriented Security Plane for Named Data Networking : Application to Content Poisoning Attack

Hoang Long Mai

Thèse
de doctorat
de l'UTT

Hoang Long MAI

Towards a Content-oriented Security Plane for Named Data Networking: Application to Content Poisoning Attack

Champ disciplinaire :
Sciences pour l'Ingénieur

2020TROY0022 **Année 2020**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

EN SCIENCES POUR L'INGENIEUR

Spécialité : SYSTEMES SOCIOTECHNIQUES

présentée et soutenue par

Hoang Long MAI

le 25 novembre 2020

Towards a Content-oriented Security Plane for Named Data Networking: Application to Content Poisoning Attack

JURY

M. Hichem SNOUSSI	PROFESSEUR DES UNIVERSITES	Président
Mme Gabi Dreo RODOSEK	PROFESSUR	Rapporteure
M. Thierry TURLETTI	DIRECTEUR DE RECHERCHE CNRS	Rapporteur
M. Paul THERON	DOCTEUR CHERCHEUR SENIOR	Examinateur
M. Guillaume DOYEN	MAITRE DE CONFERENCES	Directeur de thèse
M. Olivier FESTOR	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Personnalités invitées

M. Wissam MALLOULI	INGENIEUR SENIOR R&D
M. Edgardo MONTES DE OCA	PDG - MONTIMAGE

Abstract

Nowadays, the Internet has evolved far from its initial design. Motivated by growing change requirements to its core, the Information-Centric Networking (ICN) paradigm is proposed as a clean-slate approach for the Future Internet to address this issue. Named Data Networking (NDN) is the most mature proposal of ICN architectures. However, a novel approach also requires a significant effort to deploy, manage, and secure it. In this thesis, we proposed a content-oriented security plane for NDN and chose Content Poisoning Attack as the use-case to evaluate our proposal. In this context, firstly, we have proposed a micro detector which indicates if a metric is shifted from its normal behaviors. Moreover, we also proposed a correlation engine using the Bayesian Network that gathers the alerts from all metrics and performs the inference to predict if a router is under attacked. Secondly, we have proposed a content-oriented orchestration for the security of NDN. For this purpose, we refined the TOSCA profiles for the NDN network by extending base nodes to add NDN properties. Moreover, typical NDN security policies are added to TOSCA policies for security or performance incidents. Finally, we propose a content-oriented security plan to monitor NDN. More specifically, security functions can be distributed in Named Function Networking (NFN) which is a content-oriented network, which allows distributing the computation task in the network. We use the anomaly detection by Bayesian Network as a use-case to demonstrate that the proposed approach can improve the performance.

Keywords: Computer networks, Anomaly detection (Computer security), Bayesian statistical decision theory, Virtual computer systems

Résumé

Internet a évolué et s'est éloigné de sa conception initiale. Le paradigme de réseau orienté contenu (ICN) est proposé comme une approche pour résoudre ce problème. Named Data Networking (NDN) est la proposition la plus mature des architectures ICN. Cependant, elle nécessite encore un effort considérable pour la déployer, la gérer et la sécuriser. Dans cette thèse, nous proposons un plan de sécurité orienté contenu pour les réseaux NDN et avons choisi l'attaque par empoisonnement de contenu CPA comme cas d'utilisation pour évaluer notre proposition. Premièrement, nous avons proposé un micro-détecteur pour déceler les déviations de métriques par rapport à leurs comportements normaux et un moteur de corrélation utilisant les réseaux bayésiens afin de combiner les micro-alertes et détecter

l'occurrence d'attaques. Deuxièmement, nous avons proposé une orchestration orientée contenu pour la sécurité de NDN. Nous avons étendu les profils TOSCA en étendant les nœuds de base pour ajouter des propriétés NDN et avons ajouté des politiques de sécurité NDN pour répondre aux incidents de sécurité ou de performance. Enfin, nous proposons un plan de contrôle de sécurité orienté contenu pour surveiller NDN. Plus spécifiquement, les fonctions de sécurité peuvent être distribuées dans Named Function Networking qui est un réseau orienté contenu et permet de répartir la tâche de calcul dans le réseau. Nous utilisons le réseau bayésien comme cas d'utilisation pour démontrer que l'approche proposée peut améliorer les performances.

Mots clé: Réseaux d'ordinateurs, Détection des anomalies (informatique), Statistique bayésienne, Systèmes virtuels (informatique).

To my parents, my brother,
the reason for who I become today.
Thanks for your unlimited support and continous care.

To my loving girlfriend, Khoa Nguyen,
whose love and endless support have enriched my soul and inspired me to
complete this thesis.

Acknowledgements

This work has been carried out within Montimage France, the Autonomous Networks Environments (ERA) at the University of Technology of Troyes (UTT), the Institut National de Recherche en Informatique et en Automatique (Inria) at Lorraine Research Laboratory in Computer Science and its Applications (Loria). It is partially funded by the French National Research Agency (ANR), DOCTOR project.

This work has been accomplished under the supervision of Mr. Guillaume DOYEN and Professor Olivier FESTOR. I would like to express my deepest gratitude to Mr. DOYEN for his highly professional guidance and incessant support. He has accompanied me from my master's internship and encouraged me to discover this field. I am greatly indebted to Professor FESTOR, who has assisted me with high efficiency and valuable guidance. Without their kind and patient instructions, it is impossible for me to finish this thesis.

I would like to show especially my gratitude to Mr. Edgardo MONTES DE OCA and Mr. Wissam MALLOULI for their help and valuable advice during my doctorate. Their rigors, their problem-solving skills, and their knowledge have always been helpful to me. I highly appreciate the friendly and professional environment they have created for me during my doctorate. It was my honor and pleasure to work with them.

I sincerely thank the jury: Mr. Thierry TURLETTI and Professor Gabi Dreo, for accepting to review my Ph.D. thesis. I also would like to extend my special thanks to Mr. Paul THERON and Professor Hichem SNOUSSI for agreeing to examine this thesis.

I want to express my thanks to secretaries of UTT's doctoral school, Sophie LEVEQUE, Thésèse KAZARIAN, Pascale DENIS, Isabelle LECLERCQ for their support during my time working in UTT and to my beloved colleagues, Luong NGUYEN, Vinh Hoa LA, Huu Nghia NGUYEN and Ana CAVALLI for the wonderful time we spent together at Montimage.

Finally, I owe a lot to my dad Ly, my mom Phuong, my brother Bao and my beloved Khoa Nguyen, who support, encourage and help me at every stage of my personal and professional life, and long to see this achievement come true.

Contents

List of Figures

List of Tables

List of Abbreviations

ANFIS	**A**daptive **N**euro-**F**uzzy **I**nference **S**ystem
API	**A**pplication **P**rogramming **I**nterface
AR	**A**nonymizing **R**outer
AS	**A**utonomous **S**ystem
BN	**B**ayesian **N**etwork
BNC	**B**ayesian **N**etwork **C**lassifier
CCN	**C**ontent **C**entric **N**etwork
CDN	**C**ontent **D**istribution **N**etwork
CPA	**C**ontent **P**oisoning **A**ttack
CPD	**C**onditional **P**robability **D**istribution
CPT	**C**onditional **P**robability **T**able
CS	**C**ontent **S**tore
DAG	**D**irected **A**cyclic **G**raph
DNS	**D**omain **N**ame **S**ervice
DONA	**D**ata **O**riented **N**etwork **A**rchitecture
DDoS	**D**istributed **D**enial **o**f **S**ervice
ETSI	**E**uropean **T**elecommunications **S**tandards **I**nstitute
EU	**E**uropean **U**nion
FIB	**F**orwarding **I**nformation **B**ase
FP7	Seventh **F**ramework **P**rogramme
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
HTTPS	**H**yper**T**ext **T**ransfer **P**rotocol **S**ecure
ICN	**I**nformation **C**entric **N**etwork
ICNRG	**I**nformation **C**entric **N**etwork **R**esearch **G**roup
IFA	**I**nterest **F**looding **A**ttack
IoT	**I**nternet **o**f **T**hings
IP	**I**nternet **P**rotocol
ISP	**I**nternet **S**ervice **P**rovider
JT	**J**unction **T**ree
MLE	**M**aximum **L**ikelihood **E**stimator
MMT	**M**ontimage **M**onitoring **T**ool
NACK	**N**egative **ACK**knowledge

NDN	**N**amed **D**ata **N**etworking
NDN	**N**amed **F**unction **N**etworking
NetInf	**Net**work of **Inf**ormation
NFD	**N**DN **F**orwarding **D**aemon
NLSR	**N**amed-data **L**ink **S**tate **R**outing
NRS	**N**ame **R**esolution **S**ervice
PER	**P**IT **E**xpiration **R**ate
PFA	**P**robability of **F**alse **A**larm
PIT	**P**ending of **I**nterest **T**able
PoP	**P**oint-**o**f-**P**resence
POR	**P**IT **O**ccupancy **R**ate
PSIRP	**P**ublish **S**ubscribe **I**nternet **R**outing **P**aradigm
PURSUIT	**PU**blish **SU**bscribe **I**nternet **T**echnology
ROC	**R**eceiver **O**perational **C**haracteristic
SAIL	**S**calable and **A**daptive **I**nternet so**L**ution
VE	**V**ariable **E**limination

List of Notations

$\mathcal{H}_0$	null hypothesis, i.e. no attack case
$\mathcal{H}_1$	alternative hypothesis, i.e. attack case
X_i	random variable
$Pa(X_j)$	variables/nodes of X_j
$E = e$	Evidence
$P(X_q \mid E = e)$	the conditional probability of X_q given the evidence $E = e$
$Scope[\phi]$	the scope of the factor
α	prescribed false alarm probability
σ_x^2	variance of the random variable x
ϕ	the factor
σ	elimination order

Introduction

Context

The Internet was initially invented to allow people to connect from one location to remote resources and colleagues. Throughout its exponential growth, it became an environment primarily used for distributing data. Indeed, the rise of mobile services, the evolution of usage, the growth of the Internet of Things, the generalized use of social networks and video streaming services unquestionably increase the amount of data flowing over the networks every day. Among the daily generated traffic, a noticeable percentage of data is duplicated since most of the content is distributed from a few number of providers to many end-users. Although several solutions have been proposed at different layers of the protocol stacks, such as multicast, Content Delivery Networks, or peer-to-peer solutions, the current Internet architecture was not designed to straightly allow the distribution of content on a large scale and to a huge number of users. In addition, not only do these workarounds increase the complexity of the Internet, they also generate a bunch of new problems in the functional domains of security, mobility, scalability, and quality of service.

To address the challenge, novel paradigms to replace the Internet architecture emerged. Among them, the Information-Centric Networking paradigm (ICN) was proposed to perform a shift from the traditional host-to-host communication paradigm of the Internet to a host-to-content one. Among all ICN approaches, Named Data Networking (NDN) is the most mature and adopted one.

Although NDN is a promising candidate to replace the current Internet, network operators must carefully consider the potential opportunities, revenues, and operational (OPEX) and equipment (CAPEX) costs to deploy and operate such a novel technology. Moreover, novel technology like NDN is expected to evolve and change. Therefore an essential requirement towards a future proof internet is to find a solution to deploy new network functions or protocols while reducing both OPEX and CAPEX.

A new trend in the networking area has emerged in the last few years. As server virtualization became more popular and mature in IT data centers, the constraints of

hardware-based appliances led operators to apply standard IT virtualization technologies to their networks in late 2012. These technologies are named Network Functions Virtualization (NFV). Since then, the NFV initiative has developed a great deal of interest and is actively working to develop and execute network functions entirely in industry-standard hardware applications. The virtualization of network functions over shared pools of standardized commodity hardware resources will allow to achieve greater agility, flexibility, and elasticity in network configuration and operation, leading to reduced time on the market for deploying new networking services while reducing both OPEX and CAPEX.

Problem statement

Despite the maturity of ICN, network operators are hesitant to bring such technologies into their production networks. We argue that the Network Virtualization Function (NFV) model, which allows network operators to deploy network functions in software that runs on a standard commodity server hardware, coupled with Software Defined Network (SDN) services, is an excellent candidate to support the deployment of ICN networks. Although the primary Management and Orchestration (MANO) functional blocks are now standardized by ETSI and implemented by state-of-the-art NFV frameworks, they primarily target IP-based Virtual Network Functions (VNF). Consequently, hosting non-IP VNFs, such as NDN ones, impacts the management plane and functions, thus requiring MANO evolutions. In particular, security is indispensable and needs to be implemented before considering any NDN deployment over NFV. As a consequence, researching on a secure content-oriented orchestrator for NDN is crucial.

Before deploying such an orchestration solution, network operators must be able to collect data from a distributed system efficiently as it allows the secure content-oriented orchestrator to make decisions on management and security. However, to the best of our knowledge, there is no mechanism available in NDN to collect a full list of metrics. Moreover, to help the orchestrator make decisions and perform mitigations against attack, security functions must be conceived to enable the correlattion of metrics to detect anomalies in the network. Such an anomaly detector is still lacking in NDN. These issues lead to the need for a security monitoring plane in NDN.

Security functions, especially anomaly detectors, are high-resource tasks. As they are executed periodically in the network, and current are Machine Learning or statistical approaches, they require a significant computation effort. As a consequence, a gap between the complexity of security functions and the constraint

resources of nodes has existed and needs to be filled. Hence, a distributed anomaly detection architecture is needed.

To sum up, researching on a content-oriented security plane for NDN is crucial. The expected security plane has to include:

- A security monitoring function to collect different metrics and detect anomaly behaviors;
- A secure content-oriented orchestrator to deploy, manage and secure NDN network functions;
- A distributed anomaly detection architecture.

These challenges will be addressed in this thesis.

Contributions

The objective of this thesis is to propose a Content-Oriented Security Plane for Named Data Networking (NDN) that can efficiently address existing security threads in NDN, such as Content Poisoning Attack (CPA).

As a first step, we design and implement a monitoring plane for NDN. To that aim, a list of metrics are built and selected, and for each of them, micro detectors, able to consider any abnormal variation, are designed and evaluated both theoretically and empirically. Moreover, by leveraging a Bayesian Network Classifier, we design a correlation engine to correlate the alarms from micro detectors to detect if the NDN node is under attack. CPA is our use case to validate our proposed approach. The numerical results obtained in real deployment experiments demonstrate the efficiency of the proposed monitoring plane, especially the proposed anomaly detector. However, we found that there are still rooms for the improvement of performance for the anomaly detector, which is addressed in the third part of this thesis.

The second step of an attack processing is the mitigation and reaction. For this purpose, a Content-oriented orchestration is proposed to deploy a secure NDN virtual network. The TOSCA standard is leveraged using a crafted NDN oriented extension to enable the specification of both deployment and operational behavior requirements of NDN services. We also highlight NDN-related security and performance policies to produce counter-measures against anomalies that can either come from attacks or performance incidents.

In the context of distributed system security, we also found out that current node-centric solutions induce waste of computing resources, further enlarging the

gap between the complexity of security functions and the constraint resources of nodes. Moreover, we also found out that the result of security functions is repeated over time, and these functions could also be distributed to accelerate the computational time. On the other hand, the concept of content-oriented networking naturally matches the requirements to tackle these issues. Indeed, we can benefit the concept of cache and leverage the content-oriented networking as an execution environment to support Anomaly Detection Functions efficiently. Hence we propose a content-oriented security plane to monitor NDN. More specifically, security functions can be distributed in Named Function Networking (NFN). NFN is a content-oriented network, which allows distributing the computation task in the network. We use the anomaly detection by Bayesian Network as a use-case to demonstrate that the proposed approach can improve the performance, especially if the security function is complicated and the computing resources are limited.

Document Structure

The rest of this document is organized as follows. Chapter 1 provides an overview of ICN architectures. We first explain the need for novel Internet architecture and introduce the concept of ICN. Our focus will be on the details of NDN, which is the most promising candidate among existing ICN architectures. The current state of ICN deployment efforts is also presented in the chapter by providing supporting technologies, deployment configurations, and the analysis of NDN deployment trials. Finally, we introduce state of the art on NDN security by providing an overview of attacks in ICN and argue the choice of CPA as the selected use-case for our work.

Network Anomaly detection is the first step towards a content-oriented security plane for NDN. Among the numerous existing methods for anomaly detection, Bayesian Network Classifier is the one that got most attention in the security research community. It is also the one that we chose to support anomaly detection in NDN. As a consequence, chapter 2 familiarizes the concept of Bayesian Network by providing the definition, terminology, and the core principles of Bayesian Networks. The chapter is built as a tutorial to understand the different steps, methods, and algorithms required to build and use a Bayesian Network. The tutorial is guided by a simple use case that allows us to understand the idea of Bayesian Network. Once familiar with Bayesian networks, we present Bayesian Inference as we argue that it can be efficiently used to enhance NDN's security. Moreover, by explaining in detail different Bayesian Inference algorithms, we also demonstrate that these algorithms are high-resources consumers and that improvement can be made in both consumption and computation time.

Chapter 3 presents our design of the Monitoring Plane for Named Data Networking, step by step. First and foremost, we introduce the overall architecture of the proposed monitoring plane. Afterward, we describe the NDN monitoring probe and NDN anomaly detector, which are the two components that form the proposed monitoring plane. To introduce the NDN monitoring probe, we introduce not only its structure but also the list of monitoring metrics, as the metrics are used to feed the micro-detector anomaly detector. Then, the micro-detector results are used to feed the Bayesian network to detect if an attack occurs. Finally, we assess the overall performance of the monitoring plan using the use-case CPA.

Chapter 4 focuses on the design of a secure content-oriented orchestration. There, we first present the overall architecture of the DOCTOR NFV MANO architecture. This archicture was selected in our framewrok for NDN deployment and hosts our orchestrator. Afterward, to achieve a content-oriented orchestration, an extension is proposed for NDN in TOSCA base nodes and NDN TOSCA policies. However, to enable the NDN as network function, content-aware components need to be considered. The chapter then presents in detail how these modifications and extensions perform the content-oriented orchestration. The orchestration is evaluated in the last part of the chapter to demonstrate its applicability and performance.

Due to the complexity of the security functions and the resource constraints of the nodes, security functions have to detect the anomaly effectively by controlling the use of computing resources. To tackle these issues, chapter 5 presents a content-oriented anomaly detection architecture, which offers a promising solution to reduce the computation resources and improve the scalability of the anomaly detection function. The chapter proposes a novel approach to improve the performance of Bayesian Inference in security functions by leveraging the methodology of a content-oriented network named NFN. First and foremost, the naming scheme and data structure are proposed. Afterward, we choose the most basic and standard algorithm for Exact Bayesian Inference to illustrate how we transform it in NFN functions. The performance of NFN is then presented to demonstrate the advantage and the potential of the proposed approach.

We conclude the thesis with a summary of our contributions and a set of open issues for future work.

Chapter 1

Named Data Networking - State of the Art

Contents

Among the domains of the world which have known a steady evolution in recent decades, the Internet is certainly one of the top. Indeed, although it was originally designed to transport a *Data* object from host to host, the growing demand for content in recent decades has made this concept obsolete. As a result, in late 2000, the research community introduced novel Internet paradigms, whose purpose is to replace the current Internet architecture. Among them, the Information-Centric Networking (ICN) paradigm has emerged as an efficient yet feasible approach. After a decade of research and development, ground scientific locks have been overcome, and operational implementations have been released. Among them, Named Data Networking (NDN), has established itself as the reference one.

Consequently, the research issues that are now at the core of this paradigm shifted from early design of protocols and prototypes to credible deployment ones and several approaches have been proposed to date, all addressing the fundamental question of the cohabitation with IP. Some of them leverage the SDN concept to enable the cohabitation of these two types of protocols, while others aim at isolating them by considering virtualization solutions. As a strongly coupled issue, the question of management and security of deployed ICN networks also raises.

To help envision these concepts, in this chapter, we perform a deployment-oriented state of the art of ICN. Among its related content-oriented architectures, Named-Data Networking (NDN) is the most accomplished one and it benefits from the support of a large community of researchers and industrialists. As such, we deliberately select NDN as the ICN proposal we present subsequently and for the contributions we propose. Then, the chapter presents an analysis of different solutions and technologies that are candidates for an NDN deployment, followed by an analysis of the security issues. Finally, the last part of the chapter depicts challenges for anomaly detection in NDN, thus motivating the necessity of novel dedicated monitoring and security mechanisms.

1.1 Named Data Networking Overview

In this section, we present the necessity of novel Internet architectures. We start by introducing a class of solutions, called Information Centric Networking (ICN), which emerged about fifteen years ago from the research community. Among the different proposals which have been made in this field, we especially detail Named Data Networking, which is the most mature and acknowledged one.

1.1.1 On the need for a novel Internet architecture

In the 1960s-1970s, the Internet was initially invented to allow us to connect one location to another. As a result, its design was exclusively based on host identification (i.e. IP addresses), and data was considered as an object to be transported from an host to another. However, the Internet has evolved far from these expectations. Indeed, the Internet now stands for an environment primarily used for data distribution: the rise of the Internet of Things, Social Network, and Video streaming all increase significantly the amount of data carried, all together, in a context of host mobility. The Global Internet Phenomena Report[1] indicates that about 70 percent of

[1] https://www.sandvine.com/hubfs/downloads/archive/2016-global-internet-phenomena-report-latin-america-and-north-america.pdf

total traffic in Latin America and North America are streaming video and audio including real-time entertainment contents such as Netflix, Youtube, Amazon Video, Itunes. Undoubtedly, among them, a noticeable percentage of data is duplicated as these audios and videos are distributed from a few numbers of service providers to numerous end-users. These changes pose challenges for the current architecture of the Internet and as the demand for content grew on the Internet, several solutions have been proposed.

Firstly, the IP multicast protocols [2] all aims at preventing the useless duplication of IP packets by especially enabling the sender to send data only once, which will, if necessary, be duplicated according to a distribution tree by the network equipment in order to be distributed to several users. Nevertheless, multicast could not be deployed at the Internet level, especially for economic and political reasons between operators [3].

Secondly, Content Delivery Networks (CDNs) [4] was proposed to reduce server load and reduce the distance between users and servers. CDNs propose to replicate content available on the original server in several places on the Internet, on servers of their own. By leveraging traffic engineering solutions such as anycast routing, location-aware DNS resolution or HTTP re-directions, the requests from users are routed to the nearest CDN server instead of to the original server. However, despite the actual and massive adoption of this class of solution over years, it does fundamentally not provide an efficient Internet performance.

Finally, Peer-to-Peer (P2P) networks [5] have also been proposed to improve the sharing of large content on the Internet. In P2P networks, each network entity, called peer, act both as a client and a server and can transmit data to other users as well as to download. Although P2P networks have been deployed as overlay networks, they propose the early principle that have been then been exploited in ICN: Content Addressable Networks (CAN) [6] in which routing is achieved on the basis of a content identifier instead of a host, and the network (i.e. topology and routing tables) is organized to support a well-operating packet forwarding.

1.1.2 Information Centric Networking

Information-Centric Networking [7] (ICN) is a networking architecture performing a shift from the traditional host-to-host communication paradigm of the Internet to a host-to-content one. As such, ICN is more suitable for massive content diffusion, such as video delivery, which is one of the major use cases in the current Internet. ICN identifies each content by a unique name instead of using an IP address and it also caches content across the network for a more direct access and less binding

between the provider and consumers. In other words, the location of the information is not taken into account in ICN: while host-centric networks characterize an IP architecture on an end-to-end connection principle, Information-centric networking is featured by the correspondence between the need of a user and the content itself, regardless of the location or form of this content.

In this novel approach, content is at the heart of the communications. When a client requests a content name, the content will be forwarded to the nodes that can provide it. Once a node that can provide this data is reached, it returns it to the client, together with a signature to guarantee its integrity. Since all the forwarding entities of ICN own a storage capacity to cache the data, the latter is used to satisfy further queries with the same content name that passes through it. This principle is reminiscent of P2P networks where content will eventually be spread closer to users or be replicated at multiple points in the network, avoiding the traditional bottleneck of a single server. Briefly said, any entity in the network which stores the content can respond to a request for that content.

In terms of security, an ICN node must guarantee that any content has not been altered by an attacker (i.e. integrity). Moreover, that has indeed been published by a legitime publisher (i.e. authentication). These features imply that any node must be able to self-certify any content without the need to use a third-party certificate authority. To that aim, a content provider associates the hash of content with its name and it also embeds the public key initially generated for the data and signs this hash with the corresponding private key.

The three principal features of the ICN approach are Naming, Resolution - *Data* routing, and Caching.

- Naming: Naming schemes in ICN can be categorized into two types: flat and hierarchical [8].
  - Flat: A flat name is usually built on a mix between a cryptographic hash function of the data and the data identifier.
  - Hierarchical: A hierarchical name is similar to that of Fully Qualified Domain Names (FQDN) on the Internet for host identification and that of files on file systems, all together being components of Uniform Resources Locators (URL).
- Name resolution and data routing: Regarding Name resolution and data routing in ICN, there are two models: coupled and decoupled.
  - Coupled: In the coupled model, each node in the network handles at the same time two tasks names resolution and data routing.

  - Decoupled: In the decoupled model, different entities take responsibility for each task.

- Caching In general, ICN has two types of caching: on-path and off-path.
  - On-path: the network makes use of cached copies along the path taken by the name resolution request.
  - Off-path: the network exploits cached copies located outside the path of the name resolution request.

1.1.3 Named Data Networking

Among all ICN approaches, NDN (Named Data Networking) [9], which follows its predecessor Content Centric Networking (CCN) [10], is the most mature and adopted one.

Naming

In order to enable *Data* providers and consumers to operate on NDN names, publishers and users must agree on a naming convention. To that aim, NDN proposes a user-friendly and well-organized naming scheme which leverages hierarchical names, which is similar to URL: in NDN, each content name contains various components separated by the slash character. The hierarchical organization in NDN allows not only offering a user-friendly naming scheme but also facilitating the routing, as each route could corresponds to a level in the hierarchical name. A *Data* packet cannot be changed after being published, so a proper versioning protocol must be implemented to allow a provider to publish a new version of the content. Therefore, the first part of the name contains a unique global name, and the second part allows versioning. In addition, some content may be too large to be transported in a single packet; it must be segmented into several packets. Consequently, the last part of a packet is reserved for segmentation. The following figure illustrates an example of an NDN content name.

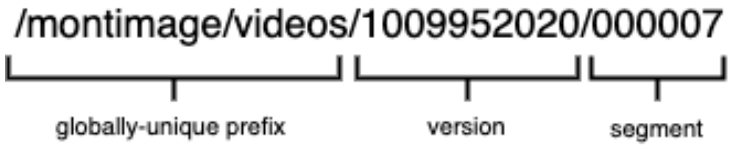


FIGURE 1.1: An example of content name in NDN

Packet

The NDN architecture employs a pull-based mechanism, using two kinds of packets, *Interest* and *Data*, which stand for a content request and response packet, respectively. Figure 1.2 illustrates the different fields they exhibit. An *Interest* packet consists of the name of the requested content, a nonce, and some guiders (the scope, the lifetime of the *Interest*). A *Data* packet represents a response to an *Interest*, which contains the requested information object. This object includes the content name, some meta-information, the content, and a signature on the entire packet (the name, content, and signed information).

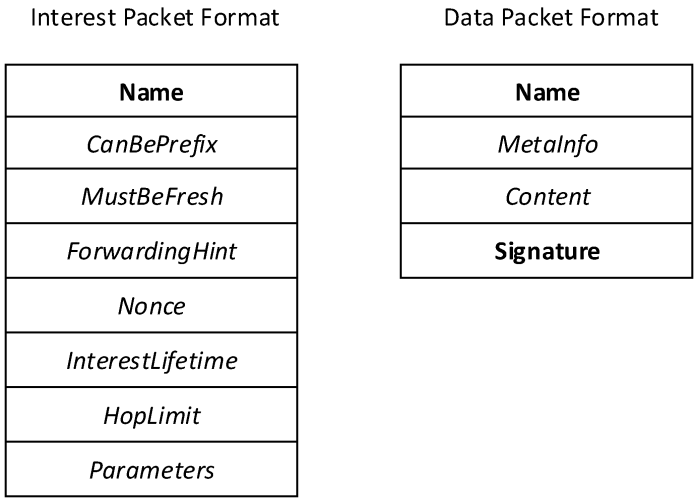


FIGURE 1.2: NDN's packets. [1]

- *Interest* **packet:**

More specifically, an *Interest* packet is a Type-Length-Value (TLV) packet defined as follows:

```
Interest = INTEREST-TYPE TLV-LENGTH
           Name
           [CanBePrefix]
           [MustBeFresh]
           [ForwardingHint]
           [Nonce]
           [InterestLifetime]
           [HopLimit]
           [ApplicationParameters [InterestSignature]]
```

- Name: the only required element in an *Interest* packet which stands for the name of *Interest*.

- CanBePrefix: This field is optional, if it is not present the Name field is an exact and fullname of a requested *Data*, if it is present, the name can be only a prefix.
- MustBeFresh: the element indicates whether a content store may satisfy the *Interest* with a cache or not.
- The ForwardingHint: the element contains a list of name delegations. Each delegation implies that the requested *Data* packet can be retrieved by forwarding the *Interest* along the delegation path.
- Nonce: it carries a randomly-generated 4-octet long byte-string. The combination of Name and Nonce should uniquely identify an *Interest* packet. This is used to detect looping *Interests*.
- InterestLifetime: it indicates the (approximate) time remaining before the *Interest* times out.
- HopLimit: it indicates the number of hops the *Interest* is allowed to be forwarded.
- ApplicationParameters: it can carry any arbitrary *Data* that parameterizes the request for *Data*.

- **Data packet:**

NDN *Data* packet is TLV defined as follows:

```
Data = DATA-TYPE TLV-LENGTH
        Name
        [MetaInfo]
        [Content]
        DataSignature
```

- Name: stands for the name of *Data*.
- MetaInfo: contains meta infos such as:
  - ContentType: stands for the type of content such as: *Data*, NACK or Key
  - FreshnessPeriod: it indicates how long a node should wait after the arrival of this *Data* before marking it "non-fresh".
  - FinalBlockId: identifies the final block in a sequence of fragments.
- The Content element can carry any arbitrary sequence of bytes.

To retrieve *Data*, a consumer puts its name in an *Interest* packet and it sends it in the network. Routers use this name to search for a *Data* provider. Once the packet reaches an NDN node that has the requested content, the routers send the *Data* packet back to the consumer following the opposite direction of the path taken by the *Interest*.

Components of an NDN node

NDN proposes a coupled model, which means that each NDN node consists of different components to handle not only the name resolution but also *Data* routing or caching. Figure 1.3 illustrates the internal structure of an NDN router. An NDN node contains four components: a *Forwarding Information Base (FIB)*, a *Pending Interest Table (PIT)*, a *Content Store (CS)* and *Faces*.

- *Forwarding Information Base (FIB)*: Similar to a routing table in IP router, the *FIB* maintains routing information to forward *Interests* thanks to a list of entries. Each entry consists of a name prefix and a list of outgoing *Faces* that can be used to send *Interests* to content providers.
- *Pending Interest Table (PIT)*: The *PIT* keeps all pending *Interests* and their arrival *Faces* into entries. Each *PIT* entry contains a name prefix, a list of incoming *Faces* of pending *Interests* for that prefix. This information is used to forward *Data* packets back to their requesters. The entry is removed when the matching *Data* is received or when a timeout occurs.
- *Content Store (CS)*: The *CS* is used for caching *Data* packets, thus providing on-path caching to improve the delivery performance in NDN. Each entry in Content Store contains the *Data* packet, a flag indicating whether the *Data* packet is unsolicited and the timestamp at which the cached *Data* becomes stale.
- *Faces*: *Faces* are the interfaces of an NDN router which allow it to communicate with other NDN nodes. Each *Face* has an identifier, which is used by the FIB and PIT to forward *Interest* and *Data* packets, respectively.

Name resolution and Data Routing

NDN uses a coupled model for name resolution and data routing. This means that each NDN router integrates components to handle name resolution and data routing at the same time.

Figure 1.4 especially illustrates the NDN router's operation for an incoming *Interest* and incoming *Data*. When an *Interest* arrives, the *Interest* packet is processed by

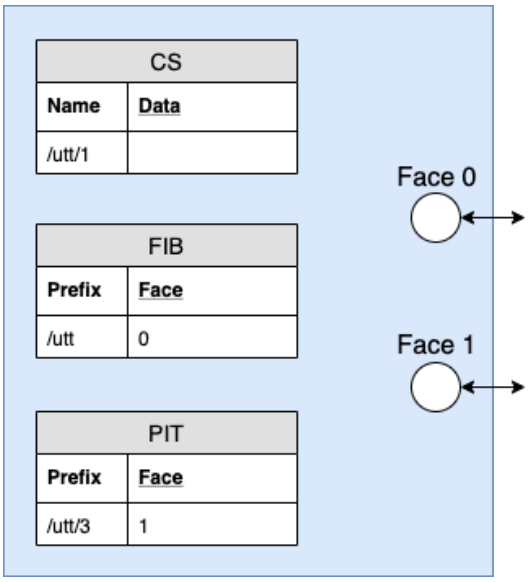


FIGURE 1.3: The internal structure of an NDN router

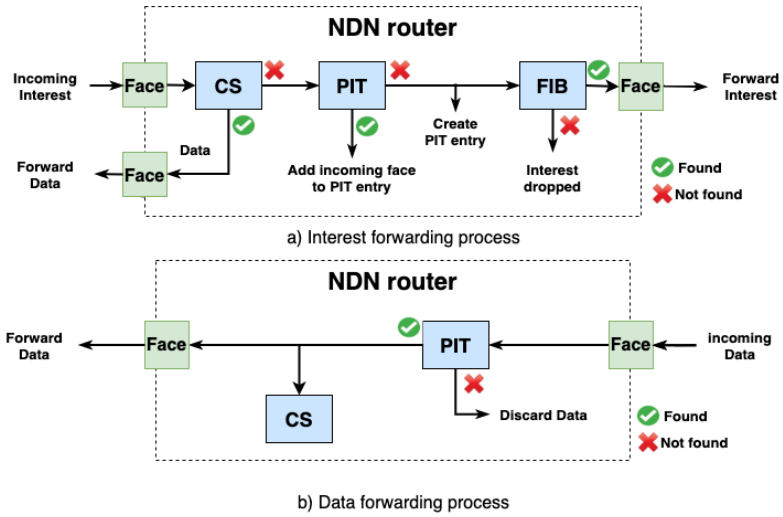


FIGURE 1.4: NDN router operations for (A) an incoming *Interest* and (B) an incoming Data

the NDN's components in the following order: the CS, the PIT, and the FIB. Firstly, if a cached copy matches, which means that *Data* is found in the CS, the NDN router immediately forwards the cached copy to the incoming Face of the *Interest*. However, when the CS does not hold any copy, the *Interest* is processed by the PIT. The latter verifies if a PIT entry corresponds to the requested content. On the one hand, if an entry matches, the interface of the *Interest* is added to the PIT, and the *Interest* is dropped since it means that a previous one has already been received and forwarded, and is eventually pending for its corresponding *Data* packet. On the other hand, if no entry matches the *Interest*, the latter is added into the PIT and is forwarded to the FIB. Finally, once an *Interest* arrives to the FIB, if no matching entry is found, the *Interest* will be dropped. Otherwise, the matching entry guides the NDN

router to forward the *Interest* to the correspond Face. Following the same procedure for each node receiving the *Interest*, the latter gradually reaches a data provider.

Regarding the backward forwarding, when a *Data* packet arrives, the NDN router checks if there is a PIT entry that matches the incoming *Data*. As each entry in PIT has a lifetime, it can expire before the arrival of the *Data* packet and be removed from the PIT, hence, making the incoming *Data* packet unsolicited. In this case, the incoming *Data* packet is discarded. On the other hand, when the incoming *Data* matches an entry in PIT, the corresponding Faces, which have been stored in the PIT entry one the *Interest* arrival, are used to forward the *Data* to downstream nodes. Moreover, a copy of *Data* is cached in the CS to satisfy further requests for the same content name. Again, following the same procedure for each node receiving a *Data* packet, the latter arrives at the client.

Security

The *SignatureInfo* and *SignatureValue* fields in *Data* packet enable the verification of data authenticity and integrity. When an NDN router receives a *Data* packet, it can retrieve the publisher's public key with the information provided in the SignatureInfo sub-field. Afterward, the NDN router uses it to decrypt the SignatureValue. The result is then compared to the hash over all other *Data*'s fields. This process allows verifying the integrity of the *Data* packet received.

Other ICN architectures

The scientific community has also proposed other ICN architectures as part of several international projects. Data-Oriented Network Architecture (DONA) [11] is to our knowledge the very first content-oriented architecture. It relies mainly on the use of new content naming mechanisms since it always uses the TCP/IP architecture. The DONA architecture still relies on the current architecture of the Internet but brings it a new name resolution that will allow distributing content effectively. The Network of Information (NetInf) [12] proposal is a research area of the Scalable and Adaptive Internet Solutions (SAIL) project [13]. It especially stands for a network architecture that can be robustly and reliably deployed across the Internet. Its purpose is to be able to connect several different technologies so that they can be deployed gradually. The Publish Subscribe Internet Technology (PURSUIT) architecture [14], which follows its predecessor Publish-Subscribe Interest Routing Paradigm (PSIRP) [15], was proposed by the European research project of the same name (FP7 EU PURSUIT). This architecture uses a publication-subscription model to deliver the contents.

1.2 Deployment of Named Data Networking

After years of research and development, NDN is now mature enough to consider a deployment in reality. However, the deployment itself of such a novel architecture, which overall aims at replacing the core of the current Internet architecture, induces technical barriers that need to be solved. In this section, firstly, we present the candidate technologies that can ease the deployment NDN. Secondly, different ICN deployment configurations are introduced. Finally, different projects and research efforts to deploy NDN are presented.

1.2.1 Supporting Technologies

Despite the promising features of ICN for the Future Internet, its deployment and especially the cohabitation with IP appears as a major lock as it potentially requires to upgrade all network components. It is commonly admitted that NDN will not replace IP in a one-shot phase, but that the deployment will rather be progressive and Software Defined Networking (SDN) and Network Function Virtualization (NFV) are two key technologies that allow it.

Software Defined Networking

The concept of the Software Defined Networking (SDN) [16] consists in separating the control plane from the data plane. A remote controller controls and configures the forwarding network component. As a result, packets are still transferred by the network component but on the basis of the instructions provided by the controller. Among the proposal of SDN implementation, OpenFlow (OF) [17] is the most successful SDN implementation. As shown in Figure 1.5, SDN comprises two essential elements: the OpenFlow Switch (OF Switch) and the Controller.

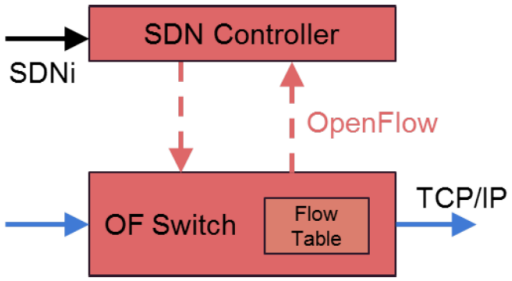


FIGURE 1.5: Simplified architecture of SDN

An OF switch contains one or more flow tables (FTs), and it communicates with a controller via the OF protocol. FTs consist of flow entries, which determine how

packets belonging to flow will be processed and forwarded. Each flow entry consists of (1) match fields (ingress port OF Switch, MAC address, IP address, ToS) to match incoming packets, (2) counters to collect statistics, and (3) a set of instructions (actions) to be applied upon a match to handle matching packets as shown in Figure 1.6.

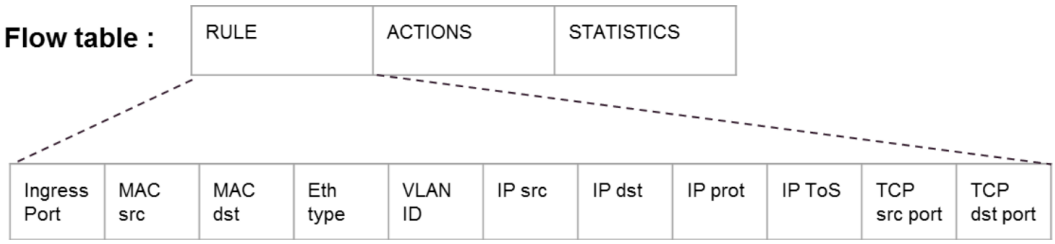


FIGURE 1.6: Flow table structure within an OpenFlow switch

When a packet arrives, if a matching entry is found in the FT, the switch applies the actions to the matched flow entry. Otherwise, it uses the table-miss flow entry to handle packets: dropping, continuing the matching process in next FT, or forwarding the packet to the controller. The latter can add, update, or delete flow entries from the switch's FTs. On the other hand, the controller can communicate with other controllers using Inter SDN Controller Communication (SDNi) [18].

Network Function Virtualization

In the past years, the development of the Network Function Virtualization (NFV) [19] paradigm is upsetting the way network functions (such as forwarding, filtering and load balancing) are deployed and managed. NFV consists in leveraging virtualization technologies to enable the deployment of network functions on commodity hardware. As such, it also stands for a deployment enabler for any novel networking function or networking paradigm such as ICN. In a traditional network, each network function is a physically instantiated and performed by a specific equipment, more or less voluminous, depending on the technology and capacity considered. On the other hand, Virtual Network Functions (VNFs) are deployed and executed by the same physical equipment, hence reducing the operating expense (OPEX) as well as the capital expenditure (CAPEX), and eventually bringing agility to network operations.

1.2.2 ICN deployment configurations

Despite the promising features of ICN for the Future Internet, its deployment and especially the cohabitation with IP appears as a major lock. The ICN Research Group

(ICNRG), chartered by the Internet Engineering Task Force (IETF), divides ICN deployment efforts into four options, namely clean-slate ICN, ICN-as-a-Slice, ICN-as-an-Overlay, ICN-as-an-Underlay [20]. The clean-slate ICN stands for an approach that aims at replacing the current IP architecture completely with an ICN one. Since this option potentially requires to upgrade all network components of the current IP architecture, to date, it is not considered as a possible option for a massive deployment. Rather, it can be considered for small scale and well-identified domains.

Besides, the 5G next-generation architecture [21] leverages NFV and SDN technologies to provide the flexibility to deploy ICN-as-a-Slice. However, to the best of our knowledge, this option has still not been experimented yet.

Hence, the two main options that are potential candidates to deploy an ICN approach are ICN-as-an-Overlay and ICN-as-an-Underlay. An ICN-as-an-Overlay approach is an approach where the ICN network is running on top of the IP protocol. It is also called a tunneling approach. Its goal is to enable communication among several ICN islands, achieved by a tunnel over the Internet Protocol. In ICN-as-an-Underlay, the NDN network integrates the existing IP network by deploying application layer gateways at appropriate locations. The introduction of NDN in given islands allows reaping the benefits of native ICN without changing the current IP architecture.

1.2.3 Analysis of NDN deployment trials

In this section, we summarize possible deployment options of NDN, along with several NDN deployment trials from different projects across all over the world. First and foremost, deployment trials that leverages SDN are presented, followed by other projects that leverage NFV to deploy NDN.

Leveraging SDN

Many research efforts argue for ICN as an overlay by leveraging the network programmability offered by SDN.

GreenICN [22] In [22], Vanlenkamp et al. propose a novel architecture, called GreenICN, to use a dedicated UDP or TCP port to identify the ICN protocol and to extend an SDN controller with an ICN module. In GreenICN, OF switches are used as the intermediate nodes to forward packets between CCN nodes. These OF switches have neither name resolution nor *Data* routing function for NDN packets. Hence, the centralized SDN controller handles these functionalities. Moreover, it

discovers the network topology, calculates the path, and reconfigures the OF Switch in order to forward the NDN packet through the OF domain.

CONET - OFELIA [23] In [24] Salsano et al. have proposed a specific ICN framework called CONET - Content Centric Inter-Networking Architecture [23]. CONET is implemented in a framework named OFELIA [25] for deploying ICN functionalities over SDN using the IP option header to the server as the name field for ICN. CONET is based on the concepts of CCN/NDN, but it follows the decoupled model for name resolution and *Data* routing, which separates the forwarding and routing operations into two planes: the control plane and the data plane. The control plane includes a Name Routing System (NRS) Nodes, which instructs a set of ICN nodes. According to the authors, this approach can reduce the cost of processing as compared to performing name-based routing in each node. The proposed architecture is naturally compatible with the SDN architecture, which also separates the control plane and the data plane. However, due to the problem of compatibility with current hardware, the OF protocol was extended in order to support ICN related operations [26, 27].

Ndnflow [28] Meanwhile, Ndnflow [28] proposes and implements an ICN module in the SDN controller to process the forwarding path computation for NDN flows, separately from the IP flows. The proposed architecture introduces a particular communication channel for ICN in parallel with the existing OF communication channel. For this purpose, a CCNx module has been added to the SDN controller to communicate with ICN-enabled switches and compute CCN-specific path. Moreover, an application-specific communication channel is created in order to ensure the communication between with CCN node. To this aim, the CCNx node is modified by integrating an SDN plugin to set up the connection with the ICN module in the SDN controller. Finally, in order to forward the CCNx packet, they use the OF protocol to modify the FT in OF Switch. On the arrival of an *Interest*, if no forwarding rule matches the requested content, the CCNx node requests guidance from the ICN Module installed in the SDN controller. In its turn, the ICN module calculates the path to forward the packet and configures the OF Switches by using OF protocol. In terms of protocol implementation, they use the application layer to exchange ICN information, which has been encapsulated in a TCP/IP packet.

Inria's solution [29] Nguyen et al. [29] implement an intermediate layer between the CCN node and the OpenFlow switch, called a Wrapper. Every packet forwarded to the OF switch is passed to the Wrapper. In more detail, on the arrival of a packet,

the OF Switch sets the in_port value as the ToS value and forwards to the Wrapper. When OF Switch receives a packet from the Wrapper, it forwards to the port corresponding to the ToS value (e.g., ToS = 2 => forward to port 2). In its turn, the Wrapper maps a face of FIB in the CCN node to the port of OF Switch using the ToS value, and in addition, it hashes the content name into IP address. In other words, it translates the FIB into Forwarding Table and vice versa. Every packet has to be handled by the Wrapper; undoubtedly, the main drawback of this solution is the forwarding performance, which is noticeably reduced. Therefore, in the controller side, ICN modules are implemented to handle the tasks for *Data* routing and caching [30]. A list of the most popular content is established by a module called Measurement Module. The Optimization module provides the optimal solution that is the set of content names and nodes caching this content. Finally, the Deflection module sets up a mapping between the content name and the nodes where content is cached. This mapping is installed using the OF Protocol.

Leveraging NFV

NFV instead argues for the separation of the IP and ICN protocols by leveraging the isolation property of virtualization. Firstly, an NFV approach for NDN must be able to define and recognize NDN services naturally and straightforwardly. Secondly, like any approach to deploy NDN networks, the final goal is to ease and shorten the deployment line and reduce the management costs of NDN services. Thirdly, to be able to be adopted by any network suppliers, the proposed approach needs to follow the current standards and be simple enough to interoperate with existing frameworks.

vICN [31] Sardara et al. [31,32] propose vICN (Virtualized ICN) to provide a flexible unified framework for ICN, which includes several functions such as monitoring. vICN consists of Linux Containers, which are connected via virtual Ethernet interfaces. vICN implements its SDN controller designed to set up ICN (via Ethernet, TCP, or UDP) and IP address and routing. It also sets up an IP-based control network using a virtual bridge, providing internet connectivity to the containers. That connectivity is used to install missing applications and report monitoring information about the network. In the containers, relevant ICN applications are started and linked with the ICN forwarder.

DOCTOR's Architecture The DOCTOR's architecture matches the requirements for an NFV approach for NDN. The proposed architecture rigorously follows the

ETSI reference architecture specification [33]. For the *Network Functions Virtualization Infrastructure (NFVI)*, it considers Docker. The choice to embrace Docker is motivated by its efficiency and large adoption. As the NDN packet format is almost totally different from the IP packet as it does not contain the same metadata, it cannot be transported by standard IP traffic. As a consequence, the current *NFVI* is agnostic to the NDN traffic; hence, the proposed architecture considers VXLAN as the encapsulation solution to carry both NDN and IP traffic. On the other hand, the proposed architecture does not need to extend other components of MANO, such as the *Virtual Infrastructure Manager (VIM)* as it controls and manages only the *NFVI* compute, storage, and network resources, which are not changed in an NDN network. Therefore, Docker Swarm is considered as the *Virtual Infrastructure Manager (VIM)* as a ready-to-use technology.

Several solutions have been proposed to aim at a practical deployment of ICN with SDN and NFV, however, to the best of our knowledge, DOCTOR [34, 35] is the first approach which pushes the content-oriented paradigm of ICN up to high-level orchestration templates. As a consequence, we choose DOCTOR as the virtualized architecture in our works.

1.3 Named Data Networking Attacks

Deployment is essential, but unquestionably not the only task which needs to be addressed. The security of such a novel architecture is also vital, as its communication model and newly introduced router components expose the network to novel security issues. To understand the threats and attacks that need to be addressed in NDN, in this section, we present an analysis of NDN's attacks. Comprehensive surveys on NDN attacks can be found in [36, 37].

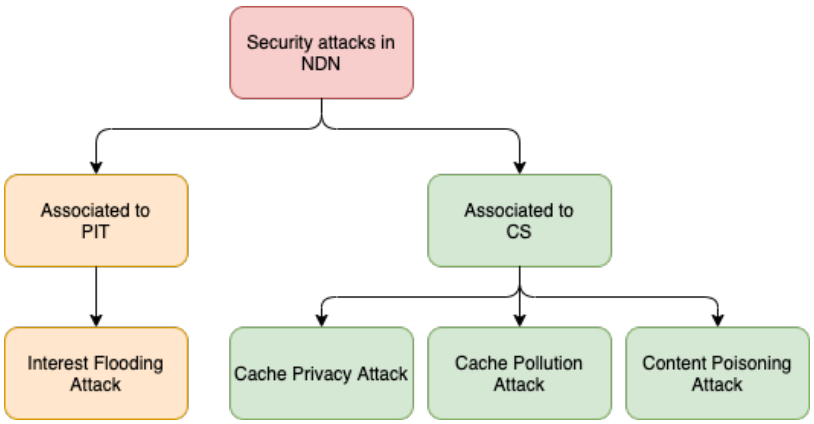


FIGURE 1.7: Taxonomy of security attacks in NDN

NDN's attacks are associated with its internal components. To the best of our knowledge, the two principal components associated with NDN's attacks are PIT

and CS. The taxonomy of these attacks is illustrated in Figure 1.7. Regarding the attacks associated with PIT, *Interest Flooding Attack* (IFA) [38] is the one that gets attention from the research community. Meanwhile, regarding attacks associated with CS, three different cache relevant attacks are considered: (1) cache pollution [39]; (2) cache privacy [40–42]; and the *Content Poisoning Attack (CPA)* [43]. FIB is not mentioned in our analysis as we argue that this component belongs to the control plane, which is hardly be compromised in practice.

1.3.1 Interest Flooding Attack

In the current Internet architecture, Distributed Denial of Service (DDoS) [44] attacks are a critical problem. The goal of DDoS is to overload the victim server by a large number of packets from a large number of end-users. The change from a host-centric to content-centric of NDN can effectively eliminate these DDoS attacks, as end-users cannot attempt to overload a specific server, as they request desired *Data* by sending *Interest* packets, and the network delivers *Data* packets upon request only. However, an NDN network can be subject to a new type of DDoS attack, namely the *Interest Flooding Attack (IFA)* [38]. IFA shares the same purpose of the DDoS attack, which consists of consuming the network resources and make them unavailable for legitimate consumers. This is achieved by sending a large number of *Interest* packets. These *Interest* packets create PIT entries in all the NDN routers from source to destination. Hence, after a while, the accumulation of PIT entries will overload the router; this phenomenon is presented in [45].

Detection

In order to detect IFA, several methods are considered, for instance, threshold-based, hypothesis-testing, entropy, Markov-based, or machine learning. Most of authors [38,46–49] propose statistical threshold-based methods. The initial idea of the method is to set an alarm for a single or multiple metrics, which raises alerts when it exceeds a threshold, as these indicators show the sign of IFA. For instance, Gasti et al. consider pending *Interest* per outgoing interface, incoming *Interest* per-interface and pending *Interest* per namespace as parameters. Meanwhile, Dai et al. consider PIT size, and finally, Compagno et al. use not only PIT size but also the ratio of incoming *Interest* and outgoing *Data* to detect IFA. On the other hand, Nguyen et al. [50] propose a hypothesis-testing statistical approach. A parametric model for the packet-loss rate in IFA is considered in this approach. This model takes the desired false alarm probability as a parameter and calculates the detection threshold. Xin et al. [51] propose a method based on entropy [52] to detect the change in the mean

value of entropy of names of received *Interest* packets on an interface. If the entropy of the interface exceeds a predefined threshold, then the attack is detected. Wang et al. [53] consider PIT Occupancy Rate (POR) and PIT Rxpiration Rate (PER) as fuzzy variables Then, these variables are used by the detection of the mechanism for the detection of IFA. The attack is detected when one of the fuzzy variables becomes high. Similar to Wang et al., the authors in [54] also use POR and PER as parameters and develop a Markov model to detect IFA. Besides, Kamar et al. [55] propose different machine-learning approaches to detect IFA too.

Mitigation

Several approaches are proposed to mitigate the IFA. The two principal mechanisms to counter against IFA are pushback and traceback. The original idea of the pullback mechanism is introduced in [46]. Gasti et al. propose a countermeasure that triggers a pushback mechanism that reduces the PIT quota of the malicious namespace for a given interface. Inspired by this paper, Afanasyev et al. [38] propose three countermeasures against IFA, which are also based on applying a limit to the number of forwarded *Interest* packets for each interface. On the other hand, Dai et al. [47] exploit PIT entries to trace back malicious *Interests* to its source. The router sends a spoofed *Data* packet as a reply to the unsatisfied *Interest* packet. As a result, the *Data* packet reaches the originating router. Then, the *Data* packet applies a filter to the interface from the originating router. Finally, other countermeasures also proposed. As an instance, Li et al. propose Interest Cash [56] an application-based solution against IFA for dynamically generated content. In [57, 58], Yi et al. propose to use the Negative ACKnowledgment (NACK) packet to help routers take the initiative when there is a problem with *Interest* forwarding.

1.3.2 Cache Privacy Attack

One of the main advantages of NDN is content caching. However, this cached content can be access by an attacker. As a consequence, the attacker can exploit this content to identify who has recently demanded the content. Using several pieces of information such as the access time, the name of content, an attacker can violate the privacy of a user by exploiting this cached content. This attack is known as the Cache Privacy Attack [40]. The cache confidentiality attack does not impede communication, but the attacker can use the information obtained from this attack to his advantage.

Detection

The cache privacy attack is hard to detect as it is more flexible than other attacks and does not affect the network. The attacker only needs to request *Interest* packets to perform this attack. As a result, it is challenging to detect CPA. Few approaches are proposed to solve this challenge. The authors in [59–61] proposed approaches that detect cache privacy attacks by analyzing attackers' behavioral parameters. These parameters are the high-interest rate, the high cache rate, or repeat requests for multiple contents in a short period. However, a change of traffic can also affect these parameters, so it is difficult to distinguish from the caching privacy attack.

Mitigation

In delay-based approaches, the router applies an additional delay to the *Data* packet before replying to it. The authors in [62, 63] propose delay-based approaches. In these approaches, an additional delay to the *Data* packet is applied to the router before replying to it. Thanks to this additional delay, the router can mitigate and prevent the attacker from distinguishing between cached and not cached *Data*. Moreover, the attacker cannot make any assumptions about content correlation, network topology, or users. However, this approach mitigates cache privacy attacks by maintaining a trade-off between privacy and the positive effect of caching on content retrieval delay. Besides, named caching-based approaches are also proposed, in which, the caching strategy is changed to protect privacy. Chaabane et al. [62] propose to allow the router to decide whenever to cache or collaborate with nearby caches. Meanwhile, Abani and al. [64] propose a caching based on betweenness centrality. Finally, the authors in [65] suggested disabling the scope field and exclude the field. They argue that an attacker can set a specific value to the fields and make cache privacy attacks easy to perform; hence, disabling these fields can prevent the attacker from performing the attack.

1.3.3 Cache Pollution Attacks

As each router has a limitation resource, its cache capacity is also limited. To exploit this vulnerability, an attacker can force to store content that is useless to degrade the caching's benefits for legitimate users; this attack is called cache pollution attack [39].

Detection

Threshold-based is the most common approach to detect the content pollution attack. Similar to threshold-based methods in IFA, its fundamental idea is to gather a set of metrics that are changed in the attack and set a limitation for them. Whenever these metrics excess the limitations, we consider that the content is polluted. Conti et al. [66] propose a lightweight threshold-based detection against cache pollution attack based on machine learning. They define a metric which depends on the frequency of content belongs to sampling, and the probability of occurrence of content to detect the attack. If the metric is higher than the threshold, the attack is detected. Based on the same methodology but based on other metrics, Guo et al., in [67], exploit the diversity of the *Interest* traversing paths within an Internet service provider's point-of-presence network to profile the attacker's strategy to populate the network. In addition, Xu et al. [43] assume that the attacker in the cache pollution attack requests a large number of *Interest* packets for a small group of unpopular namespaces. Based on that assumption, they consider distinct *Interest* packets having a common namespace and use the Monte Carlo hypothesis test [68] to compute the detection threshold. Besides threshold-based methods, other methods are proposed, such as Karami et al. [69] which propose a cache replacement policy based on the Adaptive Neuro-Fuzzy Inference System (ANFIS) or Park et al. [70] which propose an entropy-based approach that comes under the probabilistic approach.

Mitigation

Unlikely to the diversity of methods to detect the content pollution attack, only two ways to mitigate the attack are considered: proactive and reactive. In the proactive method, the main idea is to prevent the attack without a detection mechanism. Xie et al. in [39] propose a mechanism named CacheShield. CacheShield takes the responsibility to decide whenever the content is cached or not. Once a CS receives a content, if CacheShield allows, the content is cached. If not, a counter for this content is increased. If the same content object is requested again, it also increases the corresponding counter. Based on the updated counter, CacheShield re-evaluates to cache or not the content. Meanwhile, in the reactive method, the idea is more straightforward than the proactive one; once the content is detected as a polluted content, then it is removed from the CS.

1.3.4 Content Poisoning Attack

Among the most significant threats, the *Content Poisoning Attack (CPA)* [43] has been identified as a major one by the NDN community. In *CPA*, an attacker will inject

malicious *Data* into the router's cache, in order to resolve legitimate *Interests*. *Cache Pollution Attack* aims at increasing the popularity of rarely used content to force their caching by NDN routers. As a result, the caching functionality of NDN routers becomes inefficient. Meanwhile, *CPA* leverages NDN in-network caches to spread bad *Data* packets to as many users as possible. The attacker is likely to forge poisonous *Data* packets with popular content names to increase the attack's impact.

Detection

Proposed solutions to detect and mitigate CPA are restricted in number. They can be divided into two categories, namely (1) verification lessening based and (2) feedback (or exclusion) based. For routers, a naive solution to mitigate CPA consists in checking every *Data* packet signature before forwarding it. However, in reality, this is impractical due to the expensive computation cost at line speed [71]. Therefore, the goal of the first category is to reduce such verification load on routers by changing the router's verification routine [72–75], or by changing the caching policy [76] to improve the resiliency to CPA.

On the other hand, the solutions in the second category exploit the fact that a user can leverage the *Exclude* field to avoid unwanted bad *Data* packets by adapting the forwarding strategy [77] or prioritizing the contents that are least excluded [71]. It is noteworthy that most of the previous works on this topic are based on simulated CPA.

Mitigation

Most of the mitigation approaches for the content poisoning attack have used some fields of *Interest* or *Data* packets that are not included in current NDN packet specification, i.e., exclude or Publisher Public Key Digest (PPKD). The exclude field is a field in the *Interest* packet which is used by the consumer to exclude the content which it does not want as a reply. By exploiting this field, in [78], Ghali et al. propose that the router ranks its cached *Data* based on users' exclusions. A content excluded by multiple customers has a low rank, and is considered as poisoned content. Both the *Interest* and the *Data* packets have a PPKD field. It includes the hash of the producers' public key. Hu et al. [79] propose to use the PPKD field to mitigate the content poisoning attack. In the usual process of content retrieval, the user asks for content by sending *Interest* packet with the name corresponding PPKD. The user verifies its signature on receiving the content. If it is a malicious content, the user will resend the *Interest* packet, excluding the name of the poisoned content. Although

content poisoning attack can easily be excluded by performing a signature verification, such an operation is not enforced by intermediate nodes since it generates an important CPU and memory overload, which prevents the nodes from operating at line speed.

The analysis of NDN attacks shows us several attacks are possible in such a novel architecture like NDN, and among them, CPA is identified by the NDN community as one of the most significant ones. The characterization of CPA also shows that the proposed solutions to detect and mitigate the attack are not robust in terms of performance and generability. An efficient solution that can detect not only CPA but also can be adapted to detect any attack in NDN is still missing. As a result, we deliberately choose the CPA as a use-case in our work.

1.4 Conclusion

In this chapter, we highlighted the need for a new Internet architecture to adapt to the change of current Internet usage. Among the proposed candidates, Information-Centric Networking, specifically Named Data Networking, is the most promising one. NDN is considered as the replacement for the current IP architecture. However, such an immense change cannot be performed immediately. It must be done progressively, and SDN and NFV are the keys technologies to unlock the co-habitation between IP and NDN. Several deployment trials have also been presented in this chapter. Afterward, we found that DOCTOR architecture is the most compatible with current ETSI standards, thus being the best candidate to support the co-habitation between IP and NDN. In addition, such a novel architecture like NDN also brings new attacks. We performed a comprehensive analysis of NDN attacks and found out that CPA is a major threat among them. As a result, in our work, we will choose DOCTOR architecture as the solution to deploy NDN and consider CPA as of use-case.

Chapter 2

Bayesian Network Classifier - State of the Art

Contents

In our daily lives, we also often observe events to predict another event. For example, from seeing the grass is wet, one can say there is a high chance of it was raining. However, others can say that maybe the grass is wet because of a sprinkler was enabled. These predictions can be described as inferences.

As the human can make inferences, so do a computer. The aim of machine learning will be observing the current environment, calculating the probability of what could happen based on inference, choosing the scenario with the highest probability. One of the applications of inference is anomaly prediction. We can observe several metrics and events, which is a set of metrics with specific values or interval of values. Based on these values we can detect if an abnormal event is occurring and perform mitigate actions based on that.

The prediction of an event is already not an easy problem; giving an exact or approximate probability is even more challenging. Generally speaking, in this area, several approaches, such as Support-Vector Machines (SVM) [80], Neural Networks

[81], and Bayesian Network Classifiers (BNC) [82], are commonly used to tackle this issue. Among them, Bayesian Network Classifiers are one of the most considered ones.

In a Bayesian network, the fundamental idea is putting different possible scenarios and their inferences into a well-defined network, called Bayesian Network. Based on this network, we can then make inferences using various algorithms such as Variable Elimination or Junction Tree.

In this chapter, to help envision the concept of BNC, we perform a state of the art of Bayesian Network Classifiers and present a tutorial that explains how to build and use a Bayesian Network, which is resumed from the book [82]. We also present the required components linked with Bayesian networks, classifiers, and inferences. Firstly, we introduce the terminologies relevant to Bayesian Networks and their core principles. Secondly, the construction of a Bayesian Network which includes structure construction, and parameter estimation is presented. Finally, Bayesian inference algorithms are depicted.

2.1 Introduction

.

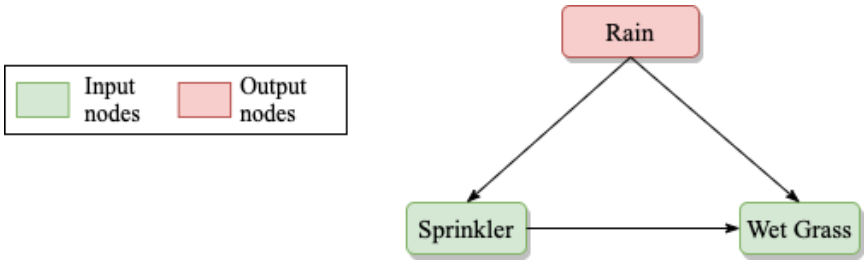


FIGURE 2.1: A simple Bayesian Network

TABLE 2.1: The data set of the RSG example

Day	Grass	Sprinkler	Rain
1	Wet	Enabled	No
2	Wet	Enabled	Yes
3	Wet	Disabled	Yes
...	...	...	...
1000	Dry	Disabled	No

To explain the concept and terminology of Bayesian Network, we use a "popular" example, which is illustrated in Figure 2.1. In this example, three events: rain,

sprinkler, and grass wet (RSG) are considered. The activation of the sprinkler or the rain has a causal relation with the grass to be wet. Moreover, in order to save water, the sprinkler is enabled only if it is not raining. Given a dataset of observed phenomena related to these three events in Table 2.1 through 1000 days providing their behaviors, on another day, if we only observe that the grass is wet, the question is to know if can we compute the likelihood that it is raining? The question is a probability problem, which can be solved thanks to Bayesian Network.

2.1.1 Terminology

Jensen et al. in [83] defines that Bayesian networks consist of (1) a set of variables and a set of directed edges between variables; (2) each variable has a finite set of mutually exclusive state; (3) the variables and directed edges form a directed acyclic graph; and (4) to each variable A with parents B1, B2 to Bn there is attached the Conditional Probabilities Distribution P(A | B1, B2 .... Bn). To understand this definition, we first introduce the terms used in Bayesian Network and its inference algorithm presented subsequently.

Variable

A random variable, called X, is a set of possible values of a random event. There are two types of variables, continuous and discretes, but we only consider discrete ones in the following section. In the context of anomaly detection, a random variable is especially an observed metric or a detection result. A random variable can take any value in a given set.

Node - Edge

In BN, each node corresponds to a unique random variable and each edge corresponds to a conditional dependency. As explained above, several events (i.e., nodes) are observable, and others are not. These observed nodes are the input to perform the inference, so they are called the input nodes. Meanwhile, the output of inference is the probabilities of different scenarios of an event that we need to predict; the corresponding node is called the output node.

In the example given above, *rain*, *sprinkler* and *grass* are random variables, called respectively R, S, G. The set of values of *rain* is $\{notrain, rain\}$, the set of values of *sprinkler* is $\{disabled, enabled\}$ and the set of values of *grass* is $\{dry, wet\}$. To simplify, we can also label each random variable with $\{0, 1\}$ values. In the context of classification, we suppose that two event, the state of the sprinkler and the status

of grass, are observed and we want to predict the state of an unobserved node, the *rain*. Hence, in this example, *sprinkler* and *grass* are input nodes, while *rain* is the output node of the Bayesian Network.

Evidence

An *evidence* $E = e$ is a subset of random variables $E = (X_{e_1}, \ldots, X_{e_m})$, where m is referred to as the number of variables in the *evidence*, representing the observed event, and $e = (x_{e_1}, \ldots, x_{e_m})$ an instantiation of these variables which represents the occurrence of these observed data. One could note that the value of an *evidence* is a subset of values from several random variables, hence, we can split an evidence into several evidences or aggregate different evidences into one.

A simple example evidence which continues the aforementioned example is "we already observed that the grass is wet today". So the evidence for this observed data is $G = 1$.

Conditional Probabilities Distribution

As in a Bayesian Network, we may know several observed data, and based on this evidence, we need to deduce the probability of other random variables in the network. The root for this deduction in probability theory is called the Conditional Probability Distribution (CPD). The CPD of Y given X is the probability distribution of Y when X is known to be a particular value, noted $P(Y|X)$.

For example, back to the RSG example, to answer the question: if we know that "the grass is wet", "what is the probability that it is raining", we must use the conditional probability distribution of R when G is known to be a particular value. Table 2.2c shows us the CPD of G when S and R are known. One may be wondering where all of these values come from. These numbers are established on the basis of the observed data. In details, the observed data gathers all instances of the three metrics during an observation period and given these instances, we can calculate the conditional probabilities. For instance, to calculate the conditional probability of G when we know the value of S is 1 and the value of R is 1, we count all instances in the observed data set 2.1 that includes the combination (S=1,R=1). Here, the number of instances which includes the combination $(S = 1, R = 1)$ is 100 and, among them only 1 instance includes $G = 0$ while 99 other instances include $G = 1$. Thereupon, the conditional probability of $G = 0$ given $(S = 1, R = 1)$ is estimated to $0.01 (= 1/100)$ while the conditional probability of $G = 1$ given $(S = 1, R = 1)$ is estimated $0.99 (= 99/100)$. Beyond, the parameter estimation is a specific task in Bayesian Network and we will discuss it in the next section.

RAIN (R)	
0	1
0.2	0.8

(A) P(R)

RAIN(R)	SPRINKLER (S)	
	0	1
0	0.6	0.4
1	0.99	0.01

(B) P(S | R)

SPRINKLER(S)	RAIN(R)	GRASS (G)	
		0	1
0	0	1.0	0.0
0	1	0.2	0.8
1	0	0.1	0.9
1	1	0.01	0.99

(C) P(G | S,R)

TABLE 2.2: The conditional probability tables

Table 2.2c shows us the conditional probability of distribution of grass is wet given the value both events: the sprinkler is turned on, and it rains. As the value of the sprinkler is turned on, it is still unknown to answer the question: what is the probability of it rains given the value of the grass is wet, the relation of the sprinkler with these two events must be considered. The probability of the sprinkler is turned on given the weather is shown in Table 2.2b. Moreover, we also have the probability distribution of the event "it is raining," which is shown in Table 2.2a. From these three tables, we can finally deduce the query probability.

Factor

A general-purpose function called a $factor$ is regarded in order to quantify the affinity of two random variables. Let $(X_1 \dots X_n)$ be a set of random variables, a $factor$ ϕ is defined as a function from $Val(X_1, \dots, X_n)$ to $\mathbb{R}$. The set of variables $(X_1, \dots, X_n)$ is called the scope of the $factor$ and denoted $Scope[\phi]$ [82]. An entry is a set of values for each variable and the corresponding value of ϕ, representing the affinity between these values. The greater the ϕ value, the more these values are compatible. A Conditional Probabilities Distribution (CPD) can be considered as a particular $factor$. The only difference is that in the case of the CPD, the sum of all possible values of ϕ equals 1, thus standing for a normalized $factor$. CPDs are also used to establish the factors. One can note that the order of variables in the factor is important, a factor of $(X_1, \dots, X_n)$ is undoubtly different from the factor $(X_n, \dots, X_1)$. Tables 2.3b and 2.3c give us an example of the difference of these two factors when the order of variables in the factor is changed.

An instance of a $factor$ $\phi_3(G, S, R)$ of the RSG example is shown in Table 2.3d. The three columns on the left describe the overall joint distributions of three random variables G, S and R. While the last column on the right contains the values of ϕ

R	$\phi_1(R)$
0	0.2
1	0.8

(A) $\phi_1(R)$

S	R	$\phi_2(S,R)$
0	0	0.6
0	1	0.90
1	0	0.4
1	1	0.01

(B) $\phi_2(S,R)$

R	S	$\phi_{2*}(R,S)$
0	0	0.6
0	1	0.4
1	0	0.90
1	1	0.01

(C) $\phi_{2*}(R,S)$

G	S	R	$\phi_3(G,S,R)$
0	0	0	1.0
0	0	1	0.2
0	1	0	0.1
0	1	1	0.01
1	0	0	0.0
1	0	1	0.8
1	1	0	0.9
1	1	1	0.99

(D) $\phi_3(G,S,R)$

TABLE 2.3: Factors (a) $\phi_1(R)$; (b) $\phi_2(S,R)$; (c) $\phi_{2*}(R,S)$; (d) $\phi_3(G,S,R)$

that match each entry in the joint distribution of the $factor$. Each line in Table 2.3d represents an entry of $factor$ $\phi_3(G,S,R)$, while the value of a $factor$ is the affinity between values in the entry. $\phi(G=0,S=0,R=1)=0.2$ while $\phi(G=0,S=1,R=0)=0.1$ means that $G=0,S=0,R=1$ is likely to be twice compatible in comparison to the case where $G=0,S=1,R=0$. Table 2.3a and Table 2.3b show $factors$ $\phi_1(R)$ and $\phi_2(S,R)$.

2.1.2 Core Principles

A Bayesian Network (BN) [82] is a graphical probabilistic model consisting of nodes and edges. Each node reflects a random variable X_i and an edge from node X_i to node X_j reflects a statistical conditional dependence between the corresponding variables. As such, X_i is called a parent of X_j (i.e. $X_i \in Pa(X_j)$) and X_j is called a child of X_i. The relationship between variables is defined by the CPDs $\mathbb{P}[X_j|X_i]$ and the prior distribution of parent X_j. A Bayesian Network Classifier (BNC) is a BN used to classify one of its nodes in a finite set of values. In other words, BNC is a BN with a set of discrete random variables $X=(X_1,\ldots,X_n)$ and a set of observed data $E=e$ standing for an *evidence*, a query variable X_q, where one needs to calculate the conditional probability of $P(X_q|E=e)$. This means that the conditional probability of $P(X_q|E=e)$ is the sum of all possible combinations of values of the other variables $X_i \in X - X_q$ of the joint probability of all values X, knowing $E=e$.

Back to the RSG example, as we can see in this Bayesian Network 2.1, G is affected by R and S, while S is also influenced by R. Hence, R and S are the parent nodes of G while R is the parent node of S too. The relationship between G, R, S is defined by the CPDs $P(G|S,R)$; $P(S|R)$ and $P(R)$. Finally, the Bayesian Network also allows us to compute the conditional probability of $P(R|G=1)$ thanks to the causal relationship shown in Figure 2.1. $P(R|G=1)$ is the sum of all possible combinations of values of S of the joint probability of all values R , knowing $G=1$. As we know,

the probability of $P(R|G=1)$ is the combination of $P(R=0|G=1)$ and $P(R=1|G=1)$:

$$P(R|G=1) = \begin{cases} P(R=0|G=1) \\ P(R=1|G=1) \end{cases}$$

Moreover, as the value of S is unknown, so $P(R=0|G=1)$ is the sum of $P(R=0,S=0,G=1)$ and $P(R=0,S=1,G=1)$:

$$P(R=0|G=1) = P(R=0,S=0,G=1) + P(R=0,S=1,G=1). \tag{2.1}$$

Similarly, $P(R=1|G=1)$ is the sum of $P(R=1,S=0,G=1)$ and $P(R=1,S=1,G=1)$:

$$P(R=1|G=1) = P(R=1,S=0,G=1) + P(R=1,S=1,G=1). \tag{2.2}$$

Finally, we can get:

$$P(R|G=1) = \begin{cases} P(R=0,S=0,G=1) + P(R=0,S=1,G=1) \\ P(R=1,S=0,G=1) + P(R=1,S=1,G=1) \end{cases}$$

The algorithm used to deduce the probability is called the marginal inference and we will present it later.

2.2 Construction of a Bayesian Network

The first and most important stage in Bayesian Network Classification is the construction of the network itself. It includes two steps: the structure construction and the parameter estimation. In this section, we present the principles and related works of these two steps.

2.2.1 Structure construction

In this chapter so far, one of the main points we have assumed is that the structure of the Bayesian Network is already established. However, the structure of the Bayesian Network must be constructed and one can build its structure by relying on (1) a naive approach in which one can consider that all input nodes are independent, given the class of the output node, (2) an expert knowledge or (3) finally data itself. In the rest of the section, we consider these three approaches to construct the Bayesian Network.

Naive Bayes

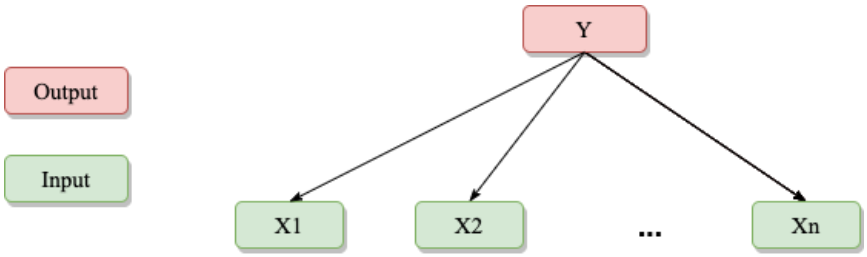


FIGURE 2.2: Naive Bayesian Network

The first choice to consider is a particular type of structure called Naive Bayes [84]. The Naive Bayes is the simplest structure of a Bayesian Network, and it does not require the need to build the structure of the network. Figure 2.2 illustrates a generic example of a Naive Bayesian Network, which includes one output node among several input nodes. The input nodes are usually observed nodes and as such they provide data to the output node to get its value. From the one hand, the output node has an impact on the input nodes. On the other hand, the input nodes are conditionally independent. Alltogether, this allows to infer the probability of the output node Y given the set of observed variables X_i.

However, Naive Bayes exhibits some limitations. The most significant drawback relies in the fact that the Naive Bayes is constructed with the strong assumption that any two nodes are independent given the value of the output node, thus motivating the reason why this approach is called "naive" Bayes. In most cases, the assumption is actually not true because, in the real world settings, the input nodes always have impacts on each other. As a consequence, the result of the classification may potentially not be accurate. Conversely, in several works such as [85,86], the authors demonstrated that even the assumption is violated, the Naive Bayes performs surprisingly well in several specific domains.

Structure construction from experts' knowledge

The construction of structure based on the expert's knowledge is the most popular and reliable approach. The basic idea is to build the causal relation between two nodes based on the expert's experience feedback. If one metric is influenced by another one, this means that there is a directed edge from this node to the influenced one. This method can be applied in various domains, especially in cyber-security [87], where experts can provide their knowledge to evaluate and establish the relations between different metrics, threats or vulnerabilities. For instance, the authors in [88] propose a signature-based activity detection based on Bayesian networks acquired from expert knowledge. A Bayesian network for predicting insider threats is

also proposed in [89], its structure being built on the research literature and expert judgment.

One of the drawbacks of the construction of Bayesian Network based on knowledge relies in its dependency to the expert. While building the structure of the network, one deliberately assume that he/she always has a completely reliable expert, and the opinions of the latter are almost identical to the real structure. However, in a real context, each one has different opinions and different points of view. As a result, the individual choices for nodes and their relations may not converge toward the most accurate solution, as experts have their different points of view and as long as their choices for a particular design of the Bayesian Network is still valid.

Structure construction from data

In some domains, the amount of knowledge required to construct the Bayesien Network is too large, thus making the manual building impossible. Besides, the relationship between nodes in the Bayesian Network can also change over time or from one use-case to another. As a result, one cannot require an expert to re-design the network at regular period. Consequently, several works related to constructing Bayesian networks from data have been proposed from the research community, and there are especially two essential ways to build such a structure: the score-based approach and the constraint-based approach.

Score-based approach

In the score-based approach, the main idea is to search through a space of candidate graphs, the best graph model that fits with the observed data set [82]. Hence, the first step is to define clearly the goal of learning and then only the learning tasks, these two steps being different.

One can identify mains learning tasks:

- **Density estimation**: its objective is to obtain the full distribution to compute the conditional probabilities in the network
- **Classification**: its goal is to use the distribution to perform a specific prediction task
- **Structure discovery**: this task is usually used to discover how nodes (e.g., metrics or variables) in the network interact with each other.

In the context of BNC, the learning goal is obviously the classification. Hence, the following discussion focuses only on the specific learning task, which tries to find the optimal model for the classification.

Unfortunately, in reality, an optimal graph model may not exist, as the number of computational operations required to obtain the optimal solution is enormous, and the provided data set is usually not enough to establish the optimal solution too. So, the actual goal is to obtain a graph model that is "close" to the real structure. There are various ways to define how "close" it is from the graph model we are searching for and the real structure. The problem now is the definition of the criteria used to evaluate the candidate graph model, or to define the "score function". Various score functions are proposed in the research community; these score functions are analyzed and compared in [90]. The authors of [91] also propose new scoring functions to score each candidate model based not only on the data but also on the experts' opinions. Meanwhile, Campos et al. [92] introduce a scoring function for learning Bayesian networks based on mutual information and conditional independence tests.

In the context of BNC, one wants to design the best model to answer the queries of the form $P(Y|X_1,..,X_n)$, given that Y is the output node, and $X_1,..,X_n$ are the input nodes. The scoring metric in this context can be the classification error, also called $0/1$ loss. Given a candidate model which defines the probability distribution P:

$$h_P(x_1,...,x_n) = \underset{y}{argmax} P(y|x_1,...,x_n) \tag{2.3}$$

is the function which produces a specific prediction of y given an evidence $(x_1,...,x_n)$. The loss function for the proposed model for each instance is defined as:

$$E_{(x_1,...,x_n,y)\ P} = [\mathbb{1}\{h_P(x_1,...,x_n) \neq y\}] \tag{2.4}$$

where, $\mathbb{1}$ is an indicator function. The function is the probability of the probability over all instances $(x_1,...,x_n,y)$ that the proposed model predicts the wrong label.

Given the score function, the next step consists in choosing a search algorithm to obtain the "best" graph model according to the score function. The two major approaches of search algorithms are local search and greedy search. In the case of local search, we initiate an empty graph. Afterward, at each step, the score function is calculated before one attempts to change the structure by adding, removing, or reversing a directed edge. If the score of the latter structure is higher than the original one, one performs the change. Otherwise, one tries with another attempt. In the case of the greedy algorithm, the first assumption that must be made is a topological order of the graph. In a topological order, a variable is sorted in after other variables, which means its set of possible parent nodes is restricted to these variables. For an instance, in the RSG example, if we consider the topological order is R, S, G; then it means, R and S are potential parent nodes of G, while R maybe the parent node of

S and R does not have any parent node. Then, we perform a greedy approach by adding the parent that maximizes the score until no improvement can be made to get the parent set for each variable.

Constraint-based approach

Constraint-based algorithms are based on the PC algorithm [93]. Recently, a modern implementation, called PC-stable [94] is also proposed. In [95], the authors summarize the algorithm as follows. First and foremost, the pairs of variables $(X_i; X_j)$ that are connected by an arc are identified. Such a pair of variables is identified based on the condition that any subset of the other variables cannot separate them. This condition is tested heuristically by performing a sequence of conditional independence tests the hypothesis that X_i is independent of X_j given the subset S of size l from the neighbors of X_i excluding X_j. The next step is to identify the v-structures among all the pairs of non-adjacent nodes X_i, and X_j with a common neighbor X_k using the separating sets found in the previous step. From now, both the skeleton and the v-structures of the network are known, the last step is then to set the remaining arc directions. More recent algorithms such as Grow-Shrink (GS) [96] and Inter-IAMB [97] proceed along similar lines but use faster heuristics to implement the first two steps.

2.2.2 Parameter Estimation

Given the structure of a Bayesian Network built either from data or experts' knowledge, learning the conditional probability between variables (namely the parameters) is a subsequent challenge. In this section we present and discuss it and especially two primary methods for parameter estimation: the maximum likelihood estimation and Bayesian estimation.

Maximum Likelihood Estimation

From the preceding discussion, we assume the factorization between an instance and the set of possibles values of a variable to get the parameter (the CPDs). Let's take the RSG example, suppose that in the 1000 days data set we observe that 800 days are rainy, and 200 days are not. Consequently, our very first suggestion is that the estimation of parameter, which describes the frequency of rainy days, is 0.8 But, the question is to know if it is actually the best estimate. Let's consider θ which is the probability that it is raining. As a consequence, the probability that it is not raining is $1 - \theta$. And the probability of the sequence to get 800 days it is raining and 200 it is

not raining is calculated as follow:

$$P(<1,0,1,1,0,..,1>:\theta) = \theta^{800}(1-\theta)^{200} \tag{2.5}$$

As we only observed a data set with 1000 different instances, the goal now is to find out the best θ to maximize the probability of the sequence 800 days raining in 1000 days. We thus define the function called the likelihood function to be: $L(\theta) = \theta^{\#rain}(1-\theta)^{\#notRain}$. Moreover, the best parameter estimation is the value that maximizes the likelihood for the sequence of 800 days, called maximum likelihood estimator (MLE). Maximizing the likelihood function can be achieved by maximizing the logarithm function of the likelihood function:

$$l(\theta) = \#rain \cdot log(\theta) + \#notRain \cdot log(1-\theta) \tag{2.6}$$

By differentiating the log-likelihood, the maximum likelihood is achieved when, the differentiation is zero, which means we get the MLE:

$$\theta_{MLE} = \frac{\#rain}{\#rain + \#notRain} \tag{2.7}$$

as expected. The simplicity of learning is one of the most convenient features of MLE. However, MLE is heavily biased in the case of a small data set. So we assume that in a larger data set, the estimation will be closer to the true value of the probability, since the more day we observe, the better the estimate.

Back to the RSG example, assume we observe the weather in 10 days and get 8 rainy days out of 10 days. A reasonable conclusion would be that the probability of rainy is 0.8. If we know that the weather is usually sunny, we believe that if we observe more and more data, for example, in 1000 days, the probability will come closer to 0 than 1? We do not want this assumption to be the absolute guide, but rather a reasonable starting point. This assumption is called prior knowledge. Now the question is how to introduce the prior knowledge to the probability distribution? Hence, another approach is proposed to tackle the problem, called Bayesian parameter estimation will be presented as follow.

Bayesian Parameter Estimation

By contrast to maximum likelihood learning, Bayesian learning explicitly models uncertainty over both the variables and the parameters. In other words, the model parameters are random variables as well.

The most important foundation of Bayesian Network is based on the Bayesian theorem [98], which is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{B} \tag{2.8}$$

Thanks to Bayesian theorem, we have:

$$P(\theta|x[1],...,x[M]) = \frac{P(x[1],...,x[M]|\theta)P(\theta)}{P(x[1],...,x[M])} \propto P(x[1],...,x[M]|\theta)P(\theta) \tag{2.9}$$

A prior distribution over the parameters, $P(\theta)$ encodes our initial beliefs. These beliefs are subjective. For example, we can choose the prior over θ for the raining event to be uniform between 0 and 1. If, however, we expect the weather to be usually sunny, the prior distribution can be peaked around $\theta = 0.1$. By applying logarithm, the parameter estimation now becomes to maximize:

$$\begin{aligned} log(P(\theta|x[1],...,x[M])) &= log(P(x[1],...,x[M]|\theta)) + log(P(\theta)) \\ &= log(\theta_{MLE}) + log(P(\theta)) \end{aligned} \tag{2.10}$$

As a consequence, the only difference between Bayesian estimation and MLE is the inclusion of the prior knowledge (i.e. the prior probability distribution $P(\theta)$). A special case is when the prior probability distribution is a uniform distribution, which means that θ is always constant whatever the data set, the posterior probability distribution in this case is just a likelihood function. Therefore, the right information will lead to good estimation, but imperfect information will lead toward a wrong estimation.

2.3 Bayesian Inference

Given the structure construction and parameter estimation discussion we provided in previous sections, we now have a complete Bayesian network to perform the classification task, which is also called Bayesian inference and in this section, we will present different algorithms that can achieve this computation [82]. To explain how the Bayesian inference works, we perform a tutorial that is resumed from chapter II in the book [82].

The *inference* designates an algorithm which consists in calculating, for each value $x_q \in Val(X_q)$, the joint distribution probability $P(X_1,...,X_n)$ and then to sum out the instantiations that are consistent with $X_q = x_q$:

$$P(X_q) = \sum_{X_1} \dots \sum_{X_i \neq X_q} \dots \sum_{X_n} P(X_1, \dots, X_n) \tag{2.11}$$

Thanks to Bayes' rule 2.8, the joint probability distribution can be expressed as follows:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i)) \tag{2.12}$$

which can be expressed in form of $factors$:

$$P(X_1, ..., X_n) = \frac{1}{Z} \prod_{i=1}^{n} \phi_i(X_i, Pa(X_i)) \tag{2.13}$$

where

$$Z = \sum_{X_1, ..., X_n} \prod_{i=1}^{n} \phi_i(X_i, Pa(X_i)) \tag{2.14}$$

Φ is denoted as the set of $factors$ in a given BN. Hence, from (2.11) and (2.13), the goal of the inference algorithm is to compute:

$$P(X_q) = \frac{1}{Z} \sum_{X_n} \phi_q \cdot (\dots (\sum_{X_2} \phi_3 \cdot (\sum_{X_1} \phi_2 \cdot \phi_1))) \tag{2.15}$$

As we can see, the key of the inference algorithm is to compute the following expression $\sum \prod_{\phi \in \Phi} \phi$. This operation is called the sum-product procedure. Calculating this expression requires significant computational resources. Therefore, to calculate it effectively, we must perform the product in a subset of $factors$.

Back to the RSG example, we have:

$$P(G, S, R) = P(R)P(S|R)P(G|S, R) \tag{2.16}$$

In other words, the conditional probability could be also expressed in form of $factors$:

$$P(G, S, R) = \frac{1}{Z} \phi_1(R)\phi_2(S, R)\phi_3(G, S, R) \tag{2.17}$$

Moreover, according to equation 2.11, we have:

$$P(R|G = 1) = \sum_{S} P(G = 1, S, R) \tag{2.18}$$

$$P(R|G = 1) = \frac{1}{Z} \sum_{S} \phi_1(R)\phi_2(S, R)\phi_3(G = 1, S, R) \tag{2.19}$$

Finally, we have:

$$P(R|G=1) = \frac{1}{Z}\sum_{S}\phi_1(R)\phi_2(S,R)\phi_3'(S,R) \tag{2.20}$$

In the context of BNC, the question is: *from this complicated expression how to get the predicted class of output node?* There are actually two ways to provide the answer: the marginal inference and the Maximum a posteriori (MAP) inference. In the marginal inference, one must calculate the expression , and get the probability for each class of output node. On the other hand, the goal of MAP inference is only to get the class the most probable of the output node. However, in the context of security, one needs not only the class of the output node but also its confidence in order to take the corresponding counter-measure or to help the decision system to make the final decision. Consequently, from now on, we discuss only the marginal inference.

There are various algorithms able to perform the marginal inference of a BN; these algorithms are classified into two groups: Exact Inference and Approximate Inference. In Exact Inference, one computes the conditional probability distribution over the variables of interest and retrieves the required conditional probability query. However, the exact inference in Bayesian networks is NP-hard [99]. Hence, in some cases to reduce the complexity, one can use approximation techniques based on statistical sampling, which is called Approximate Inference [100]. The first gives an exact result but it is extremely expensive in time and memory. The second, as for it, requires fewer resources, but the result is only an approximation of the exact solution.

2.3.1 Exact Inference

The two most common Exact Inference algorithms are Variable Elimination and Clique Tree propagation [101]. In the Variable Elimination algorithm, one eliminates variables, which are not queried and observed, one by one. Clique Tree propagation is based on Variable Elimination but caches the computation so that many variables can be queried at one time.

Variable Elimination Algorithm

The Variable Elimination (VE) algorithm, illustrated in Pseudo-code 1, is the basic way to perform a Bayesian inference. It calculates some sub-expressions and caches the result of intermediate computations to avoid generating an exponentially

high number of computation steps. The core problem of VE is to compute the sum-product, which is extremely expensive in time and memory. Moreover, in practice, due to the structure of the Bayesian network, one can cache computations that are otherwise computed exponentially many times.

S	R	$\phi_3'(S,R) = \phi_3(G=1,S,R)$
0	0	0.0
0	1	0.8
1	0	0.9
1	1	0.99

(A) $\phi_3(G=1,S,R)$

R	S	$\phi_1'(R,S) = \phi_1(R)\phi_2(S,R)$
0	0	0.12
0	1	0.08
1	0	0.792
1	1	0.008

(B) $\phi_1(R)\phi_2(S,R)$

R	S	$\phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$
1	1	0.0
1	2	0.288
2	1	0.1584
2	2	0.00198

(C) $\phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$

R	$\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$
1	0.288
2	0.16038

(D) $\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$

R	$P(R\|G=1) = \frac{1}{Z}\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$
1	0.6423
2	0.3577

(E) $\frac{1}{Z}\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$

TABLE 2.4: Factors (a) $\phi_3(G=1,S,R)$; (b) $\phi_1(R)\phi_2(S,R)$; (c) $\phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$; (d) $\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$; (e) $\frac{1}{Z}\sum_S \phi_1(R)\phi_2(S,R)\phi_3(G=1,S,R)$

More specifically, the input of the VE algorithm consists of two parts: the $factors$ and the *evidence*. The $factors$ are established by CPDs and they do not change when the algorithm performs the inference at different times, while the *evidence* $E = e$ is retrieved after each iteration of the execution. The output is the conditional probability $P(X_q|E=e)$.

Given the observed data (*evidence*) $E = e$, we perform a *factor reduction* to reduce the complexity of each $factor$ in the set of $factors$ Φ by removing the joint distributions that do not match the *evidence* (lines 1-3). Several variables remain in the query probability after the *factor reduction* as one may observe only a subset of variables $E \subset X$. These variables will be eliminated by the sum-out procedure. For this purpose, the next step consists in eliminating these variables according to an Elimination Order (line 4). For each variable in the Elimination Order, one first multiply all the $factors$ that include this variable, generating a product $factor$ (lines 6-10). Then, one sums up the value of the variable and eliminates it out of this combined $factor$, generating a new $factor$ that is entered into the set of $factors$ to be processed (line 11). Afterward, once all variables have been eliminated, the only variable which remains is the query variable X_q. At this moment, one performs once

again a *factor product* to get the final $factor$ over the distribution of X_q (lines 13-15). Finally, one normalizes the $factor$ by dividing each value by their sum (lines 16-17).

Algorithm 1 Variable Elimination algorithm

Input: *initial factors* (Φ) and *evidence* (E=e)

Output: Conditional probability $P(X_q|E = e)$

1 **foreach** $\phi_i \in \Phi$ **do**
2 $\quad \phi_i \leftarrow \phi_i(E = e)$ `// Factor reduction`
3 **end**
4 Select Elimination Order (σ);
5 **foreach** $x_i \in \sigma$ **do**
6 $\quad$ **foreach** $\phi_j \in \Phi$ **do**
7 $\quad\quad$ **if** $x_i \in Scope[\phi_j]$ **then**
8 $\quad\quad\quad \psi_i \leftarrow \psi_i * \phi_j$ `// Factor product`
9 $\quad\quad$ **end**
10 $\quad$ **end**
11 $\quad \phi_i \leftarrow \sum_{X_i} \psi_i$ `// Factor marginalization`
12 **end**
13 **foreach** $\phi \in \Phi$ **do**
14 $\quad \varphi \leftarrow \varphi * \phi$ `// Factor product`
15 **end**
16 $Z \leftarrow \sum_{X_1 \ldots X_n} \varphi$
17 $P \leftarrow \varphi / Z$ `// Factor normalization`

Back to the aforementioned example, the input $factors$ for the VE algorithm are $\phi_1(R)$; $\phi_2(S, R)$ and $\phi_3(G, S, R)$ and the *evidence* is $G = 1$. The output is the conditional probability $P(R|G = 1)$. According to the VE algorithm, firstly, we perform the $factor\ reduction$ of the $factor$ $\phi_3(G, S, R)$ given the *evidence* $G = 1$ to retrieve a new $factor$ - $\phi'_3(S, R)$ as shown in 2.4a. Accordingly, we now have three factors $\phi_1(R)$; $\phi_2(S, R)$ and $\phi'_3(S, R)$ and the query probability becomes:

$$P(R|G = 1) = \frac{1}{Z} \sum_S \phi_1(R)\phi_2(S, R)\phi'_3(S, R) \tag{2.21}$$

As R is the sole query variable, we must eliminate the variable S from the equation. Firstly, by performing the factor product between three $factors$ $\phi_1(R)$; $\phi_2(S, R)$ and $\phi'_3(S, R)$ as shown in Table 2.4b and table 2.4c, we retrieve a new $factor$ called $\phi''(R, S)$. Then we perform the $factor\ marginalization$ to get $\phi'''(R) = \sum_S \phi''(R, S)$ as shown in 2.4d. Finally, we normalize the $factor$ to get the query conditional probability $P(R|G = 1)$ as shown in 2.4e.

Clique tree

Another algorithm that is also common to perform the inference task in the Bayesian network is the Clique tree algorithm, also named Junction tree [102]. The basic idea of the Clique tree algorithm is similar to Variable Elimination, which uses factors as calculation objects through a computational operation such as factor product, factor marginalization. However, in this algorithm, we use a global data structure called the Clique tree to schedule these operations. The Clique tree is a tree of clusters of nodes that allows facilitating the variable elimination. The Clique tree is must satisfy two majors conditions:

- For each factor ϕ, there is a cluster c such that $Scope[\phi] \subseteq x_c$
- For every pair of clusters $c(i), c(j)$, every cluster on the path between $c(i), c(j)$ contains $x_c(i) \cap x_c(j)$

Consequently, for each Bayesian Network, we can generate a huge number of Clique trees. Moreover, the optimal one among these candidates trees is the one that makes the clusters as small and modular as possible. Unfortunately, the identification if such an optimal tree is an NP-hard problem [103]. Several works [104–106] are proposed to tackle the issue.

When the Clique tree is built, the first step is to assign each factor to a cluster and initialization factor followed by the factor reduction with evidence in each cluster. Now, one obtains a new tree that includes only non-observed variables. Starting from a cluster, one calculates the factor product in this cluster and sends the result, which is also a new factor, as a "message" to the next cluster; this process is called belief propagation. Finally, one performs the inference queries in the final factor to get the classification result.

Back to the RSG example, Figure 2.3 illustrates the variable elimination on a clique tree. We propose a simplified clique tree for the RSG example, which includes only two clusters: R, S and G, S, R. The first step consist in assigning the three factors $\phi_1(R), \phi_2(R|S), \phi_3(G|S, R)$ into the two clusters. The second step stands for the factor reduction, given the evidence $G = 1$. Now, one starts from cluster 1, and calculate the factor product between $\phi_1(R), \phi_2(R|S)$, then cluster 1 shares its "belief" about R and S to cluster 2 through the "message" $\phi'_1(R|S)$. Finally, in the fourth step, one takes into account the "belief" of cluster 1 and the factor $\phi'_3(S, R)$ in cluster 3 to get the final factor. From this final factor, we can deduce the classification result, in a similar way to the Variable elimination algorithm.

The exact inference algorithms are resource-consuming processes. Several research initiatives propose various ways to accelerate them. Lu et al. [107–109]

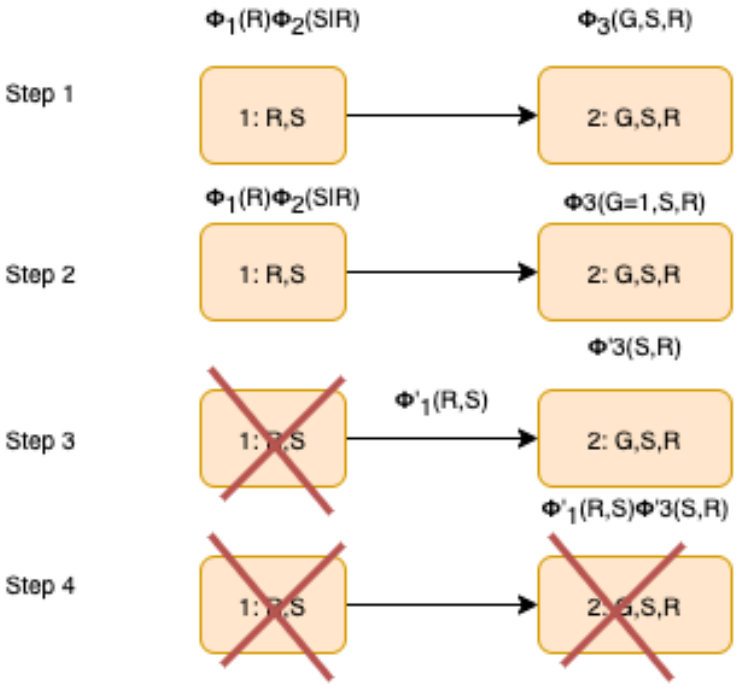


FIGURE 2.3: Simplified clique tree for the RSG example

propose a parallel Message passing approach that distributes the computation of each message passing in the inference algorithm and accelerates the algorithm using GPU. Yinglong et al. in [110–114] propose an exact inference algorithm by decomposing and merging junction trees and distributing the subset of junction trees in the network.

Both of these works demonstrate the possibility of distributing the computation effort of BNC inference, but they only focus on the parallel approach using GPU or distribute the subset of the junction tree. None of them leverages the repetition of computational operations, which can substantially reduce the computational time.

2.3.2 Approximate Inference

In the preceding analysis, we saw that the exact inference is a slow and resource-consuming process because the computational complexity of both variable elimination and clique tree are exponential. As a result, other methods are proposed to approximate inference. There exist two leading families of approximate algorithms: sampling methods and variational methods.

Sampling-based inference

Given that, for a probability distribution, we may have n possible outcomes associates with different probabilities $\theta_1, ..., \theta_n$ Il manque un morceau dans cette phrase

FIGURE 2.4: Sampling from a probability distribution

qui ne comporte pas de proposition principale.

Sampling from a probability distribution is not an easy task as we must sample from a multinomial distribution with multiple possible outcomes and associated probabilities. Usually, we perform a uniform distribution over $[0, 1]$. Hence, in the context of a probability distribution with a multinomial variable, we should subdivide a unit interval into n regions with region i having the size θ_i to reduce the sampling to a single uniform variable. We then sample uniformly from $[0, 1]$ and return the value of the region in which our sample falls. Figure 2.4 illustrates how to transform the multinomial distribution to range $[0, 1]$. The sampling from a Bayesian Network is a straightforward process called forward sampling, which samples variables in the topological order. We start by sampling the variables with no parent. We generate samples which satisfy that the probability it is raining is 0.8. Then, we sample from the next generation by conditioning these variables' CPDs to values sampled at the first step. We proceed like this until all n variables have been sampled After performing the sampling process, we can get the result for the specific prediction task. In the RSG example, the topological order is R, S, G. We start to sample from the variable R, which has no parent. Then based on these samples of R, we generates samples for s by respecting the conditional probability of S given the value of R. And finally, we generate the samples for G by respecting the conditional probability of G given S, R.

There are two major ways to perform sampling efficiently: rejection sampling and likelihood weighting. In rejection sampling, we reject the sample once an evidence variable has been sampled to take on a value which is inconsistent with the evidence of the query.

By contrast, the main idea of likelihood weighting [115] is, rather than sampling the evidence variables, forcing them to be consistent with the evidence and then reweighting the sample to account for the CPD's of the evidence variables. Other methods such as Markov Chains Monte Carlo [116] or Gibbs sampling [117] also proposed to address the challenge.

There is a major shortcoming of sampling methods is that one can not conclude if the current sampling solution is enough for the inference. Sampling methods have historically been the primary way of performing approximate inference, although, from the 2000s years, variational methods have emerged as superior alternatives.

Variational methods

The variational inference [118] is also called inference as an optimization because its goal is to cast inference as an optimization problem. In variational inference, given that we want to calculate an intractable probability distribution P, we will find the most similar distribution q which contains the same number of variables i.e.:

$$q(x) = q(x_1, x_2, ..., x_n) \tag{2.22}$$

in a set of tractable distributions, then get the prediction using the "approximate" distribution.

The problem now becomes an optimization problem in which the optimization objective is to maximize the "similarity" between the candidate distribution and the real distribution. The Kullback-Leibler (KL) divergence [119] is a famous and common tool in the field of probability and information theory which is able to measure the similarity of the "approximate" distribution and the real distribution. The next step is to narrow the searching space for the "approximate" distribution. Several distributions are candidates, such as Gaussian processes, exponential families, neural networks, or latent variable models. Among them, the most widely used distribution is simply a fully-factorized:

$$q(x) = q_1(x_1)q_2(x_2)..q_3(x_n) \tag{2.23}$$

This choice, called mean-field inference, is the most popular choice as it is easy to optimize and it works well. The key issue now is to find the best set of q_i to optimize the Kullback-Leibler (KL) divergence. The most widely used algorithm for solving this optimization problem, is the coordinate ascent variational inference [120]. The final factors q_i are good enough for performing the MAP inference; however, it is not quite equal to the real marginal distributions.

2.4 Conclusion

In this chapter, we presented the background and state of the art of Bayesian Network Classification through different steps, from the structure learning to parameters learning and finally the Bayesian Inference algorithm. As mentioned in the preceding discussion, although different choices are potential for structure construction, the most suitable one in the security context is the structure construction based on the experts' knowledge. The reason for the choice is that a security expert can add, modify metrics, vulnerabilities, or the relationship between different events,

which could be influenced when abnormal situations occur. Regarding the parameters learning, the Bayesian estimation depends on the inclusion of the prior knowledge. However, the prior knowledge is unclear for us when we detect an attack. Hence, here the safest choice for parameter estimation is still MLE. Finally, to perform the inference task, as the ultimate goal is to detect abnormal behaviors, its confidence in the prediction results is also a good indicator for the security operator to decide of any action to implement. The inference task however has accordingly, the exact inference is undoubted, the better choice. One can use either the clique tree algorithm or variable elimination algorithm as these two algorithms are based on the same computational operations. However, exact inference is a slow and resource-consuming process, where the computational operations could be repeated over time. Hence, there is still room for improvement.

Chapter 3

A Monitoring Plane for Named Data Networking

Contents

The core objective of our work is to design and build a security plane for named data networking. The first contributing element is a monitoring plane. Its aims are to (1) instrument and collect meaningful data from the entire network and its components and (2) based on the collected data, detect abnormal behaviours or attacks and notify a central orchestration component acordingly. These elements are integrated in the Monitoring Plane for Named Data Networking described in this chapter. We start with a brief motivation for such a monitoring plane. We then sketch its overall architecture followed by a detailed description of the observed metrics and the Bayesian Network Classifier for Anomaly Detection we designed. The numerical results obtained by applying the framework to a Content Poisoning Attack scenario are presented to evaluate the performance of the proposed anomaly detection system. Some conclusions are drawn on the framework and its usage for CPA detection.

The work presented in this chapter was conducted in collaboration with Ngoc-Tan Nguyen and the contributions described in the chapter were presented as a full paper in [121] and as chapter in the thesis of Ngoc-Tan Nguyen [122] .

3.1 Motivation for a Monitoring Plane for NDN

Being able to efficiently collect data from a distributed system is the first step towards its management. All management areas, including security, need to have their functions fed with meaningful data describing what is going on in the managed networks. Since NDN has its own architecture and elements (a specific naming scheme, a novel data exchange operations mode, redesigned router entities, ...), the data needed by the security functions is also specific.

To grasp the activities of information-centric networks, standard host or network centric metrics are often meaningless. A set of new metrics need to be defined and used to instrument the information-centric network entities. Composite metrics for network security analysis are proposed in [123]. Those are : the minimum number of hosts an attacker will use to compromise the target host, the average of all path lengths, etc. In the context of information-centric networks, these metrics are not relevant since they either cannot be computed nor do they represent the behavior of the network. Relevant metrics need to cover at least in-network caching and content-name routing.

As a framework, NDN also leads to new attacks and threats, some of which are still yet unknown. A solid monitoring model needs to be extensible to integrate new metrics that will be needed to capture the new attacks and threads.

To cope with these requirements, we design a flexible monitoring framework for NDN. It builds on an original functional architecture which integrates a set of dedicated metrics, a specifically designed and developed monitoring probe able to capture and analyze NDN packets and a novel anomaly detection solution, able to raise relevant alarms towards the manager.

3.2 Overall architecture of the proposed Monitoring Plane for NDN

The monitoring plane includes three basic functions: a monitoring probe, an anomaly detector and a manager. They are several options available on how these functions can be organized and placed in the network as well as in the way they interact.

For the purpose of capturing and analysing NDN packets, the monitoring probe is best installed locally on each router. It takes the responsibility to monitor the NDN traffic entering and leaving its hosting node. A set of metrics are collected and computed by the probe to feed the anomaly detector.

The organization and location of the anomaly detector in the network needs specific attention. The detector can be either centralized or distributed. It can be located in the routers or on dedicated hardware. Since the detector will operate on all monitored metrics, we assume that the traffic between the probe and the monitor will be substantial. This volume will even increase with the introduction of new metrics to cover new yet unknown attacks. To minimize overhead, we prone a joint hosting of the probe and the detector in a single host, close or within a router. This means that the detector, which acts at network level should be distributed. The distribution induces communication overhead among the detector elements but far less of that with the probes. Regarding computation overhead, one can consider the design of a coordination protocol coupled with remote computation capacities. Such a solution will be introduced in Chapter 5.

To counter attacks, counter-measures need to be selected, deployed and activated within the network. This requires the cooperation of multiple components in the network to mitigate the attack. We place this feature in the manager which will be placed at a tenant level, over-sighting the detectors in all nodes. This entity is detailed in Chapter 4.

Figure 3.1 illustrates the overall architecture of our NDN management plane which includes the data plane, the monitoring plane and the decision plane. The data plane contains virtualized components that carry and process NDN traffic. The Monitoring plane collects operational data from the network, analyses this data and does anomaly detection. The management plane reacts on alerts from the monitoring plan to trigger counter-measures in the network.

3.3 NDN Monitoring Probe

To instrument the routers, we designed a dedicated monitoring probe. This probe operates on a set of metrics carefully designed to cope with anomaly detection.

3.3.1 Structure of NDN Monitoring Probe

A probe collects meaningful data from multiple sources to feed management entities. In our case, the sources are the different functional blocks of an NDN router.

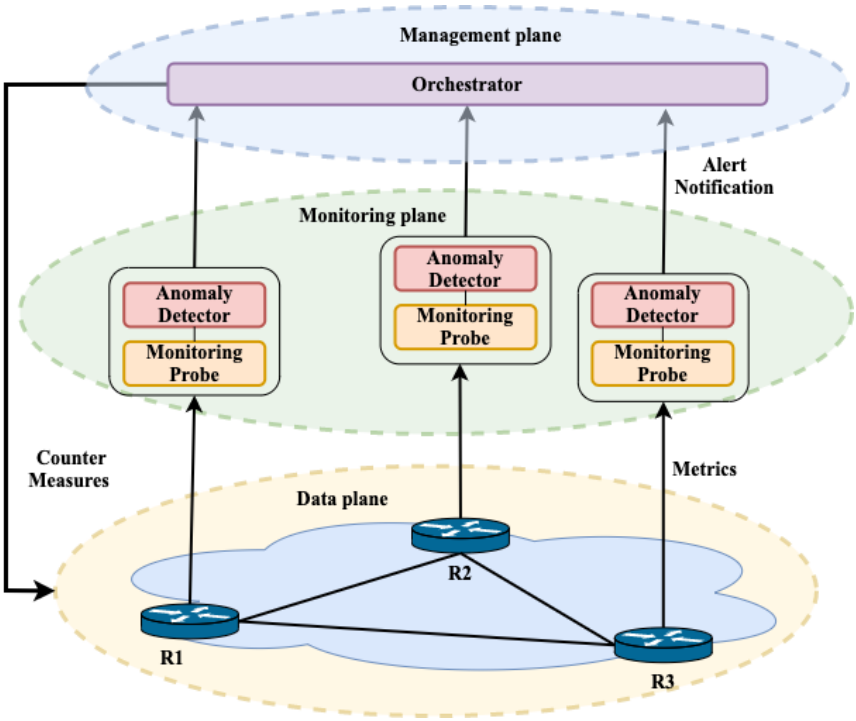


FIGURE 3.1: Overall architecture of the proposed Monitoring Plane for NDN.

These blocks are, as illustrated in figure 3.2 : CS, PIT, FIB, and Faces. For the purpose of attack and abnormal behaviors detection, all of these components should be monitored. To remain compliant with the reference implementation of NDN, we chose to use an external probe as a collector for our metrics. Therefore, the extraction of metrics is performed by the Montimage Monitoring Tool (MMT) [1], which is dedicated to the monitoring of network traffic, log files and application traces and helps various network/function/system events and metrics to be derived according to user needs. The modular plug-in based architecture of the probe enabled us to easily add support for additional protocols, applications as well as new metrics.

NDN currently offers one management protocol called the NFD Management Protocol[2]. It enables the collection of data related to the status of an NDN node, but the list of supported metrics is poor and not extensible. To the best of our knowledge, there is no mechanism available in Information-Centric Networks to collect metrics that are unavailable in the NFD Management Protocol, such as the number of Cache hit, the number of a cache miss, or the number of packets dropped. We therefore built our own tool to collect those metrics.

As already said above, we avoid modifying the NFD implementation and instead favor the extraction of the necessary information from the NFD log. We developed a specific plugin in our probe to support this extraction. Each log entry of

[1] http://www.montimage.com/products.html
[2] http://redmine.named-data.net/projects/nfd/wiki/Management/

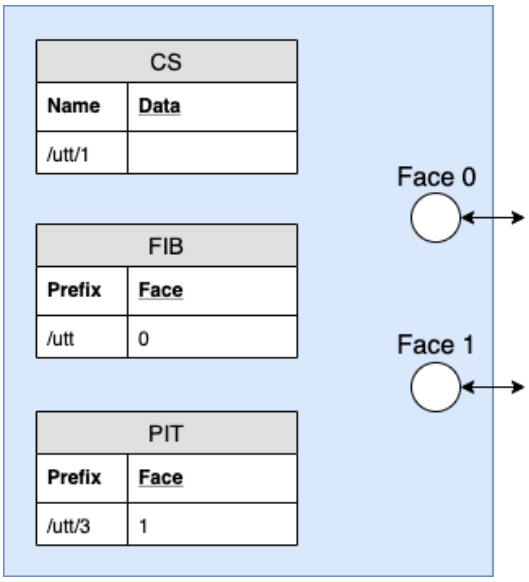


FIGURE 3.2: The internal structure of an NDN router.

the NFD log corresponds to an event in the daemon. It contains: (1) the timestamp; (2) the event name; (3) the face; and, (4) the corresponding event such as the arrival or departure of an *Interest* or a Content Store miss as illustrated in the following log. Explicit metrics can be interpreted directly from these log entries, while implicit metrics can be deduced from the log.

```
1503332255.605719 DEBUG: [Forwarder] onIncomingInterest face=264
interest=/com/good/content4
1503332255.605777 DEBUG: [Forwarder] onContentStoreMiss
interest=/com/good/content4
1503332255.605933 DEBUG: [Forwarder] onOutgoingInterest face=265
interest=/com/good/content4
```

The collected metrics are encapsulated in report packets sent to the local anomaly detector. The log encapsulation is illustrated in Figure 3.3.

Structure	LENGTH	N_ATTRIBUTES	TIME	Protocol NDN	Field ID METRIC	Length	Value	Protocol NDN	Field METRIC VALUE	Length	Value	Protocol METADATA	Field PROBE ID	Length	Value
Example	64	3	1492079399.171466	631	6	1	1	631	7	1	10	1	6	8	4

FIGURE 3.3: The proposed monitoring packet.

The encapsulated metrics are sent to the anomaly detector via an event bus. The main motivation for an event bus is to avoid redundancy in the sent reports. In fact, the anomaly detector is not the only component that receives the metrics collected by the monitoring probe. A dashboard called Operator is also using these metrics to help operators to monitor the network states over a web interface. Figure 3.4

illustrates the detailed structure of an NDN monitoring probe and NDN anomaly detector installed in an NDN router.

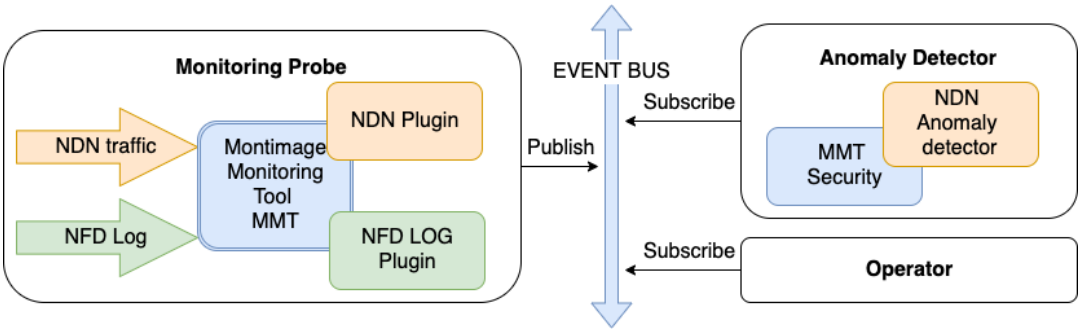


FIGURE 3.4: The structure of monitoring components.

3.3.2 Monitoring Metrics

To provide insights into the current state of a computer network, a monitoring plane needs to gather as much information as possible on the operational state of an NDN node. Since no standard metrics exist for NDN monitoring, apart from those of the NFD Management Protocol, we define a comprehensive set of metrics to reveal any potential events in the network. To establish this set of metrics, we have considered all relevant components inside an NDN node, including (1) the *Faces*; (2) the *Content Store* (CS); (3) the *Pending Interest Table* (PIT) and (4) the *Forwarding Information Base* (FIB). For each component, we applied the Case diagrams[3] approach to instrumentation.

Faces are the interfaces of an NDN router over which NDN packets are sent and received. The numbers of incoming and outgoing packets in a sampling period, including *In Interest*, *In Data*, *In NACK*, *Out Interest*, *Out Data*, *Out NACK*, are natural metrics to describe the components activity and additional characteristics, such as the volume, the frequency, the correlation between requests (*Interest*) and contents (*Data*) can be easily inferred from these metrics.

Together with the number of incoming and outgoing packets, the number of dropped packets (i.e. *Drop Interest*, *Drop Data*, *Drop NACK*) is also considered since the router can drop packets according to its strategy. These metrics are critical to detect abnormal events in NDN operations as it is unusual for nodes drop packets.

On the arrival of an *Interest*, the requested content is checked in the Content Store of the NDN router. This operation ends in either a cache miss or a cache hit.

[3]This refers to the model defined by Jeffrey Case to systematically build MIB objects for IP-based protocols

Based on the limited capacity of the cache and cache replacement policy, the CS decides to insert new valid *Data* into its storage or delete the cached data when it is complete. Since content removal from the cache is inevitably the product of adding new content to the same cache, we agree not to track removal.

To summarize, for the CS, we monitor the number of occurrences of the miss (*CS Miss*), hit (*CS Hit*) and insert (*CS Insert*) events during a specified interval.

The PIT repesents a table in which an NDN router tracks valid *Interest* it forwarded and which were not yet satisfied by the network. These entries are used to reverse-path forward *Data* packets. Similarly to CS, the number of entries created (*PIT Create*), deleted (*PIT Delete*), updated (*PIT Update*) as well as the current number of entries (*PIT Number*) are of interest to be tracked regularly from the PIT. Besides, an *Interest* has a lifetime period which stands for the time it remains stored in the PIT before either the arrival of a *Data* or before being removed by the router. An *Interest* lifetime expiration describes an abnormal situation which also needs to be monitored for both performance and security purpose. We aggregate the number of *PIT Unsatisfied* in any node.

A simple counter-measure to lifetime expiration of PIT entries consists in increasing the *Interest* lifetime, making it stay in the PIT longer. This is a typical example of a bad solution since the counter-measure can be exploited by an attacker to launch an attack by deliberately increasing this lifetime and making the PIT saturated quickly. To highlight this type of malicious behavior, we created the *PIT Exist Time* metric which holds the average of each value considered in the sampling period to feature the existing time of entries in PIT.

We intentionally decided not to include the FIB in our metrics chart. Unlike the other elements included in our selected set, the FIB belongs to the control plane, and its modifications are attributed only to static routing configurations or routing protocol updates. Either way, its metrics are less likely to change as compared to those of other components and, thus, are less useful when one needs to feature the node's behaviour promptly.
We also argue that other metrics can indirectly capture the FIB malfunctions. For instance, an increase in dropped *Interest* and *Data* can mean that the FIB is not working correctly.

Table 3.1 syntheses the proposed metric list and their description.

TABLE 3.1: List of metrics in an NDN node

	Metric	Description
Faces	*In Interest*	Periodic number of incoming *Interest*
	In Data	Periodic number of incoming *Data*
	In NACK	Periodic number of incoming *NACK*
	Out Interest	Periodic number of outgoing *Interest*
	Out Data	Periodic number of outgoing *Data*
	Out NACK	Periodic number of outgoing *NACK*
	Drop Interest	Periodic number of dropped *Interest*
	Drop Data	Periodic number of dropped *Data*
	Drop NACK	Periodic number of dropped *NACK*
CS	*CS Insert*	Periodic number of insert in CS
	CS Miss	Periodic number of Cache miss in CS
	CS Hit	Periodic number of Cache hit in CS
PIT	*PIT Create*	Periodic number of PIT entries created
	PIT Update	Periodic number of updates in PIT
	PIT Delete	Periodic number of PIT entries deleted
	PIT Unsatisfied	Periodic number of PIT entries unsatisfied
	PIT Number	Current number of PIT entries
	PIT Exist Time	Average of PIT entries' existing time

3.4 NDN Anomaly Detection

To efficiently deal with complex events, a correlation engine capable of collecting events needs to be designed. The detection solution must be capable of identifying any deviant behaviour of a node. To that aim, in this section, we present generic micro-detector for all the metrics presented above and we integrate their results in a correlation engine based on a Bayesian Network.

3.4.1 Micro Detector

An anomaly occurrence can be detected through the observation of any aberration of the metrics listed above. However, different metrics may change in various ranges of values. Since we chose a BN approach to detect anomalies, a direct integration of all metric types will increase the complexity and computation cost for estimating parameters and inference in the BN. Consequently, to be able to adapt to the natural dynamic aspect of network traffic and to minimize the BN's complexity, we propose a micro-detector to detect any significant change of a metric from its normal behaviour.

We argue that controlling the false alarm probability is of great value from an operational point of view to prevent frequent false alarms and, thus, ensure robust detection of irregular events. As such, the micro-detectors presented subsequently

are built using statistical hypothesis testing theory, which allows achieving a prescribed Probability of False Alarms (PFAs). Let us denote x_i , $i = \{1, \ldots, t\}$ a metric value observed at time i. In a normal situation, x_i follows a statistical distribution $\mathcal{P}_{\theta_0}$ where θ_0 is a distribution parameter. When an abnormal event happens, the distribution parameter will change, hence, allow us to detect the situation.

For each metric, the first step consist in deciding which distribution it is following. Considering the necessity for a simplified classifier, we deliberately select the normal (Gaussian) distribution to model our metric behaviour for normal traffic as it is a well-known distribution, allowing simple parameters estimation and developing hypothesis test. Although the classifier's general purpose is for anomaly detection, the anomaly behaviour of the metrics can be diverse. Therefore, we argue that providing an accurate model for anomaly behaviours is unnecessarily complicated. Instead, we focus on modelling metrics in regular traffic and detecting whenever a metric significantly increases or decreases as compared to its normal value range, since this could be a consequence of an anomaly.

Moreover, since the mean distribution has more meaning than the variance when considering the increase or decrease of a metric, it is assumed that the variance σ^2 of the model is fixed and the statistical hypotheses will only depend on the mean. Hence, we define the hypotheses for our detection problem as follows:

$$\delta\left(x_t, \ldots x_{t-n+1}\right) \begin{cases} \mathcal{H}_0 & \text{if } \tau_1 \leq \sum_{t-n+1}^{t} x_t \leq \tau_2. \\ \mathcal{H}_1 & \text{if } \sum_{t-n+1}^{t} x_t < \tau_1, \\ \mathcal{H}_2 & \text{if } \sum_{t-n+1}^{t} x_t > \tau_2, \end{cases} \tag{3.1}$$

where n is the window size considered for the detection. Note that in (3.1), $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively correspond to significant decrease and increase with respect to the normal behavior. The thresholds τ_1 and τ_2 are established, to maintain the desired false-alarm probability, as follows:

$$\tau_1 = \Phi^{-1}\left(\alpha_0/2\right)\sqrt{n}\sigma + n\mu_0 \tag{3.2}$$

$$\tau_2 = \Phi^{-1}(1-\alpha_0/2)\sqrt{n}\sigma + n\mu_0 \tag{3.3}$$

where Φ and Φ^{-1} respectively represents the standard normal cumulative distribution function and its inverse function. α_0, is the desired PFA of the micro detector. Equations (3.2) and (3.3) show that the thresholds τ_1 and τ_2 are functions of $\alpha_0, n, \mu_0, \sigma^2$. While α_0 and n are chosen based on the requirements for the micro detector, μ_0, σ^2 can be estimated from metric's normal behavior. In short, the threshold τ can be computed in advance and guarantees the desired PFA, regardless of the metric's behavior under attack.

3.4.2 A Bayesian Network Classifier

Micro-detectors based on a single metric can only describe a specific aspect of the state of the NDN node. However, different types of attacks have different effects on the metric. Moreover, the change in a metric can not alone conclude on the situation of an NDN node. A simple example illustrating this limitation is the case when a standard increase in traffic can affect the *Interest* metric and could be confused with an attack. This is why alerts from a single micro-detector can not be interpreted as an attack on/in the NDN node. A correlation of micro-detector alerts is needed to do that. In order to demonstrate the causal relationships between micro-detectors, a Bayesian Network structure is proposed and illustrated in Figure 3.7. Its nodes correspond to the micro detector of a metric. There are two key reasons for choosing a BN. Firstly, BN can correlate a small set of metrics to detect if an anomaly event occurs. Secondly, BN is a Bayesian probabilistic method, and the measurable data of computer networking are not entirely predictable. Thus, BN can effectively deal with the underlying random nature of the observed metrics.

In our BN, the detection of anomalies that may occur in the NDN network is represented by the *Anomaly* node. Directed edges in the BN reflect the causal relationship between metric pairs, the *Anomaly* node and a metric. The edge is sketched from the node that affects the impacted node. Since the causal relationship in each BN is built on the basis of expert knowledge of the relationship between the nodes in the BN, we deliberately choose NFD forwarding pipelines to deduce the causal relationship as all metric changes occur within these pipelines. A forwarding pipeline is a series of steps that operate on a packet or a PIT entry that is caused by a specific event [124].

We group NFD pipelines in four main categories that are triggered by external factors: (1) *Incoming Interest*; (2) *Interest unsatisfied*; (3) *Incoming Data* and (4) *Incoming NACK*. It is worth noting that the actual pipelines in [124] cover many details of the NFD practical implementation. To keep the Bayesian network explanation straightforward and understandable, we deliberately simplify the pipelines to retain the most relevant information to the proposed metric list.

Incoming Interest pipelines

Figure 3.5 illustrates the *incoming Interest pipeline*. When an *Interest* packet arrives, NFD first checks if it violates any reserved prefix rule (e.g. the */localhost* prefix is reserved for internal communications between components only). If it is the case, the *Interest* packet is dropped, meaning that *In Interest* impacts *Drop Interest*. If the *Interest* packet is duplicated with one that was already registered in the PIT, NFD

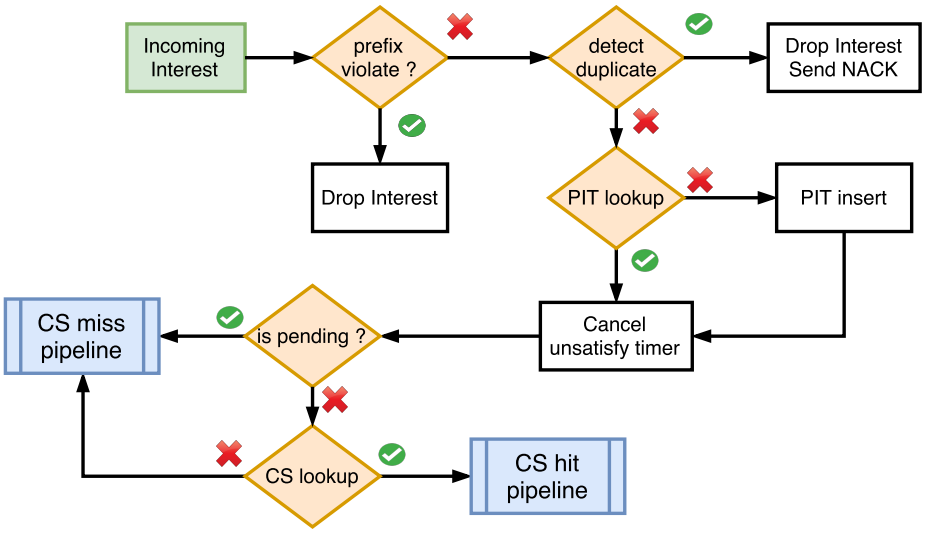


FIGURE 3.5: Incoming *Interest* pipeline

sends an *NACK* message to notify its downstream. Otherwise, NFD will insert a new PIT entry or update one that already exists by canceling the unsatisfied timer. Hence, *Out NACK*, *PIT Create*, and *PIT Update* are affected by *In Interest*. In the next step, NFD performs the CS lookup for a matching cached *Data* packet and enters the *CS Miss* or *CS Hit pipelines* accordingly, implying the influence of *In Interest* on *CS Miss* and *CS Hit*.

In case of a cache hit, the corresponding PIT entry is removed. NFD then verifies and refuses the cached *Data* packet if it violates reserved prefixes before sending it downstream. As a result, *CS Hit* impacts *PIT Delete*, *Drop Data* and *Out Data*. If there is a cache miss, the corresponding PIT entry is updated once again by adding the incoming face of the *Interest* packet and the unsatisfy timer is set. Hence, *CS Miss* also affects *PIT Update*. Besides, the *Outgoing Interest pipeline* is triggered. Once again, prefix violation will be verified before forwarding the packet to other nodes. Therefore, *Out Interest* and *Drop Interest* are influenced by the *CS Miss*.

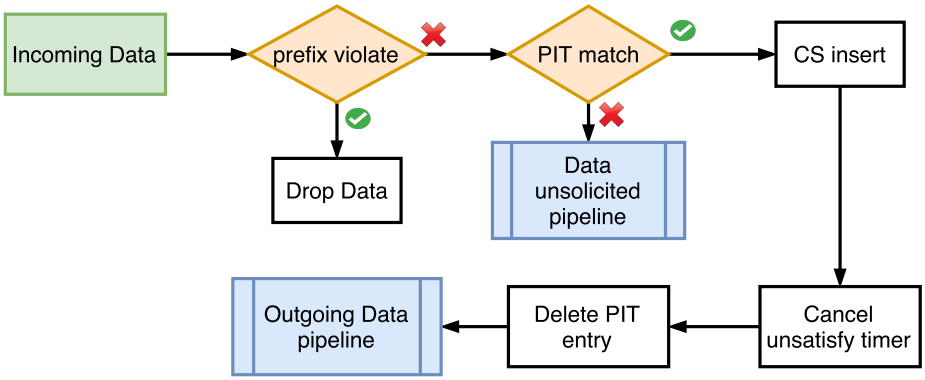


FIGURE 3.6: Incoming *Data* pipeline

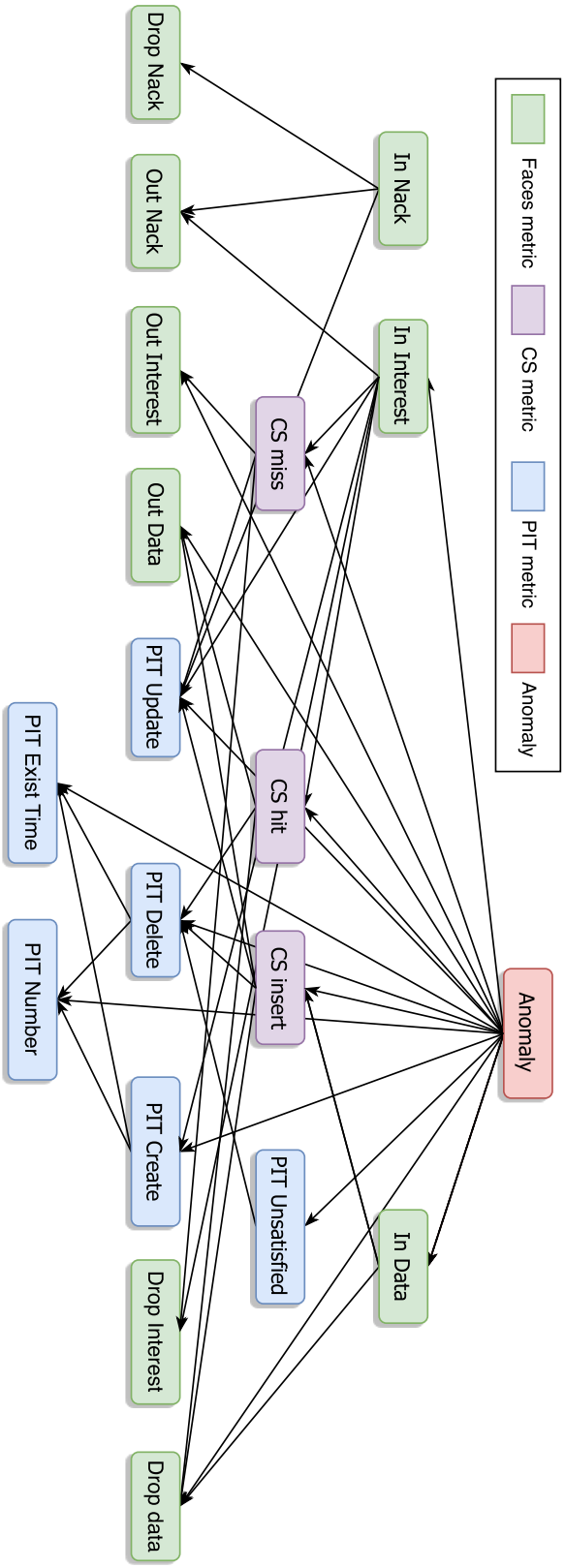


FIGURE 3.7: The proposed Bayesian Network.

Interest unsatisfied pipeline

After forwarding an *Interest*, the NDN node waits for a *Data* or *NACK* packet from the upstream, but only for a while. Each entry in the PIT has an unsatisfy timer. When this timer expires, NFD considers that no upstream node can satisfy the *Interest* and remove the entry from the PIT. Hence, the *PIT Delete* is affected by *PIT Unsatisfied*. Because *PIT Exist Time* and *PIT Number* also change whenever a PIT entry is removed or created, they are also impacted by *PIT Create* and *PIT Delete*.

Incoming Data pipeline

Figure 3.6 depicts the *incoming Data pipeline*. When a *Data* arrives, NFD first checks and drops *Data* packets that violate any reserved prefix. NFD then verifies whether the *Data* matches any PIT entry. If no matching PIT entry is found, the *Data* is considered unsolicited. Depending on the policy of NFD, unsolicited *Data* can be dropped or inserted in the CS. If a corresponding PIT entry does exist, the *Data* is also added in the CS. Note that even if the pipeline inserts the Data to the CS, whether it is stored and how long it stays in the CS is determined by CS admission and replacement policy[124]. Thus, *In Data* undoubtedly affects *Drop Data* and *CS Insert*.

When a *Data* is inserted into the CS, NFD will cancel the unsatisfied timer for each matching PIT entries, implying PIT updates. After a while, the corresponding PIT entry is deleted, and the *Data* packet is passed to the *Outgoing Data pipeline*, where NFD verifies and drops the *Data* if there is any prefix violation before it is forwarded to downstream. Thus, *CS Insert* impacts *PIT Delete*, *Drop Data*, *Out Data* as well as *PIT Update*.

Incoming NACK pipeline and Outgoing NACK pipeline

The last pipelines we mentioned in this paper are *Incoming* and *Outgoing NACK*. When NFD received a NACK, the *NACK* packet will be dropped if it's no longer a relevant entry in the PIT. Then before sending an outgoing *NACK* downstream, the corresponding in-record will be erased from the PIT, which is considered as a *PIT Update*. Therefore, the *Drop NACK*, *Out NACK* and *PIT Update* are influenced by the *In NACK*.

Finally, by observing the behaviors of metrics in cases of attack, we establish the list of metrics that change substantially when the attack happens, and as a consequence, consider that these metrics are influenced by a CPA attack and probably other attacks as well. These metrics is listed as follow: *In Interest*, *In Data*, *CS Miss*, *CS Hit*, *CS Insert*, *PIT Create*, *PIT Update*, *PIT Update*, *PIT Delete* and *Drop Data*. As these

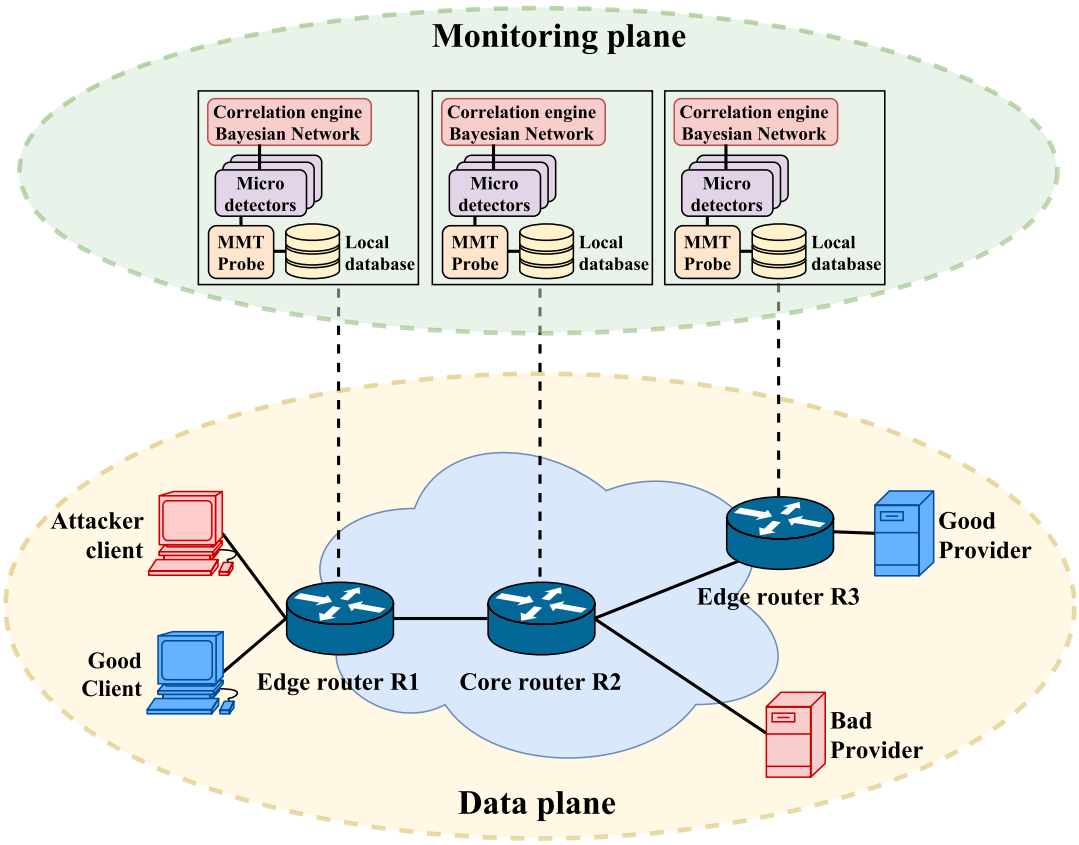


FIGURE 3.8: Use-case topology for Content Poisoning Attack.

metrics increase as compared to normal traffic when CPA happens, undoubtedly, these nodes are impacted by the Anomaly node, then, the structure of our Bayesian network is complete.

3.5 Monitoring Plane Assessment: The Case of Content Poisoning Attack

We assess the relevance of our approach in the particular context of CPA. CPA is a typical attack in NDN and has impact on multiple metrics, while normal changes in the observed network behaviour often impact only a single metric or a subset of the metrics. CPA is therefore a good candidate for validation.

3.5.1 Testbed and Scenarios

Testbed

The experimental results have been obtained within a virtual environment that reproduces the topology presented in Figure 3.8. The topology contains three routers:

an edge router on the client side R1, the core router R2 and the router R3 that represents the edge router and caching system on the legitimate provider side. Both good and bad (or attacker) clients connect to R1. In each router, an MMT probe is deployed to collect data for the 18 selected metrics. The MMT probe is used in the forwarding layer to collect data from the NFD log. The probe uses around 10% of the memory and has no significant impact on the router. The data is processed by a micro detector, which retrieves the node status from the local database. Finally, the results of the micro detectors are aggregated by the Bayesian Correlation Engine Network to determine the status of the node.

The good client behavior is implemented by reproducing a realistic web traffic pattern that considers both the popularity of the content and the statistical properties of requests over time. The attack is being carried out by dedicated bots. The payload of the attack is controlled by the amount of traffic sent by the bots and their number.

The entire infrastructure is virtualized. This offers the possibility to adjust the topology as needed. For example, adding users requires only the spawning of containers.

Scenarios

Different scenarios attack of CPA are carefully considered in [125]:

- ***Unsolicited*** : This scenario is discovered by the authors in [125], it exploits the fact that an NDN node accepts *Data* even if it comes from a different face from the one the corresponding Interest has been forwarded to. This allows a bad provider to send *Data* packets unsolicitedly with a chance to match a pending Interest in the target NDN node. When this implementation flaw is discovered, NFD only checks whether the *Data*'s name matches any existing PIT entry and ignores the case that the *Data* does not arrive at the same face to which the corresponding *Interest* has been forwarded [125].

- ***Multicast*** : is one of the forwarding strategies available in the current NFD, which forwards it to all faces registered in the corresponding FIB entry. When the router receives the bad *Interest* from bad clients, it then forwards them to both the legitimate and bad providers, according to the *multicast* forwarding strategy. Consequently, a *Data* packet is returned by both providers. However, due to the shorter delay, the bad *Data* arrives at R2 first, resolves the corresponding PIT entry and is cached in R2 and then in R1. Meanwhile, because of the longer delay, the legitimate *Data* arrives R2 later and is dropped since the PIT entry has already been resolved [125].

- ***Bestroute*** : is the default forwarding strategy used by the current NFD. Once an *Interest* is forwarded, the NDN router will suppress any similar *Interest* with the same *Name*, *Selectors* but different *Nonce* if it arrives during a *retransmission suppression interval*. The purpose of such an interval is to prevent a malicious entity from retransmitting too frequently. After this interval, a similar *Interest* received is considered as a valid retransmission and is forwarded to the next lowest-cost face that has not been previously used, hence opening the door for the bad provider to act. When all registered faces in the FIB entry have been used, it is forwarded again to the first-used face. Thanks to the collaboration with malicious clients, the attacker can generate additional similar *Interests*, forcing router R2 to use the other route towards the bad provider, hence pulling bad *Data* to caches in R2 and R1 [125].

Among the three scenarios mentioned above, the *unsolicited* scenario occurence can easily be prevented by patching NFD [125]. Indeed, since version 0.5.0 (October 4^{th}, 2016), NFD has implemented a fix that prevents unsolicited *Data* packets to be forwarded. On the opposite, while the two other scenarios (i.e., *bestroute* and *multicast*) are more tedious to implement, they can hardly be circumvented as they exploit the NDN protocol and are compliant with the NFD forwarding rules. Therefore, we focus the relevance and the efficiency evaluation of the general-purpose monitoring plane based on abnormal values of a comprehensive set of metrics jointly in the two latest scenarios.

3.5.2 Numerical result

To evaluate the efficiency of both the micro detectors and the monitoring plane, each CPA experiment lasts 10 minutes, the attack being launched during the last 5 minutes. As depicted in Figure 3.8, the bad provider standing for the attacker has a smaller delay than the legitimate provider but it does not take part in the default route and, therefore, it is associated with a higher cost. On the one hand, the legitimate users keep sending *Interests* until the good content is received using the *Exclude* field to avoid getting the same bad *Data*. On the other hand, the attacker clients behave the opposite way: they keep sending *Interests* excluding the good content so that the requests for the poisoned content names can be tricked to go to the bad provider. [4]

The results of the monitoring in term of CPA detection is summarized in Table 3.2. For a meaningful comparison, the legitimate clients average traffic amount

[4] At the time of the experiments, the version of the NDN Packet Format Specification used for the attack is v0.2.1. In this version, the NDN packet contains the *Exclude* field, which has been subsequently removed from version v0.3.

TABLE 3.2: Efficiency of monitoring plane with respect to CPA detection for two different scenarios and various attack payload.

Scenario	Attack rate (# Interest/s)	5	10	20	50
CPA *bestroute*	% True Positive	95	95.33	97	98.33
	% False positive	<0.01	<0.01	<0.01	<0.01
CPA *multicast*	% True Positive	63.33	72.83	79.33	96.33
	% False positive	<0.01	<0.01	<0.01	<0.01

is about 10 *Interests* per second. The attacker traffic thus ranges approximatively from half of legitimate traffic to five times the legitimate traffic. Of course, one can note from Table 3.2 that the larger the attack payload, the easier the detection. Interestingly even for a small attack payload, the monitoring plane can detect a CPA with high efficiency. It is worth noting that those results correspond to detection results for every metric measurement, without correlating those results in time which is a very powerful leverage to improve any real-time anomaly detection system. One can also note that regardless the attack payload and the scenario, the monitoring plane rarely triggers false positives; this is very important for the monitoring plane in an operational context because it avoids the pollution with numerous false alarms. These false alarms create a burden to deal with and end-up in preventing mitigating the real attacks.

Micro Detector Evaluation

Due to the diversity of the behavior of the metrics when anomalies occur, we focus on correctly modeling metrics in normal traffic. Figure 3.9 depicts the kernel estimated density function for some illustrative metrics (*In Interest*, *CS Hit*, *PIT Number*) and their approximated normal distributions. The figure shows that for most of our metrics (e.g. *In Interest* and *CS Hit*), the empirical distribution is close to the normal distribution, indicating the relevance of the model. Nevertheless, the model does not fit well for some metrics (e.g. *PIT Number*), because their value range is close to zero and the variance is narrow. However, to retain the simplicity and the reusability of the micro-detector, we deliberately accept this accuracy decrease in the modeling for this minor part of metrics and intend to compensate it by correlating other micro detectors' alarms.

Figure 3.10 illustrates the theoretical and the empirical PFA of our micro-detectors for different metrics. Each metric's threshold was normalized by the mean and standard deviation of its normal behavior so that the performance for various metrics can be demonstrated in the same figure. For most of the metrics (e.g. *CS Hit*, *In Interest*), the empirical and the theoretical PFA match closely, implying the ability to

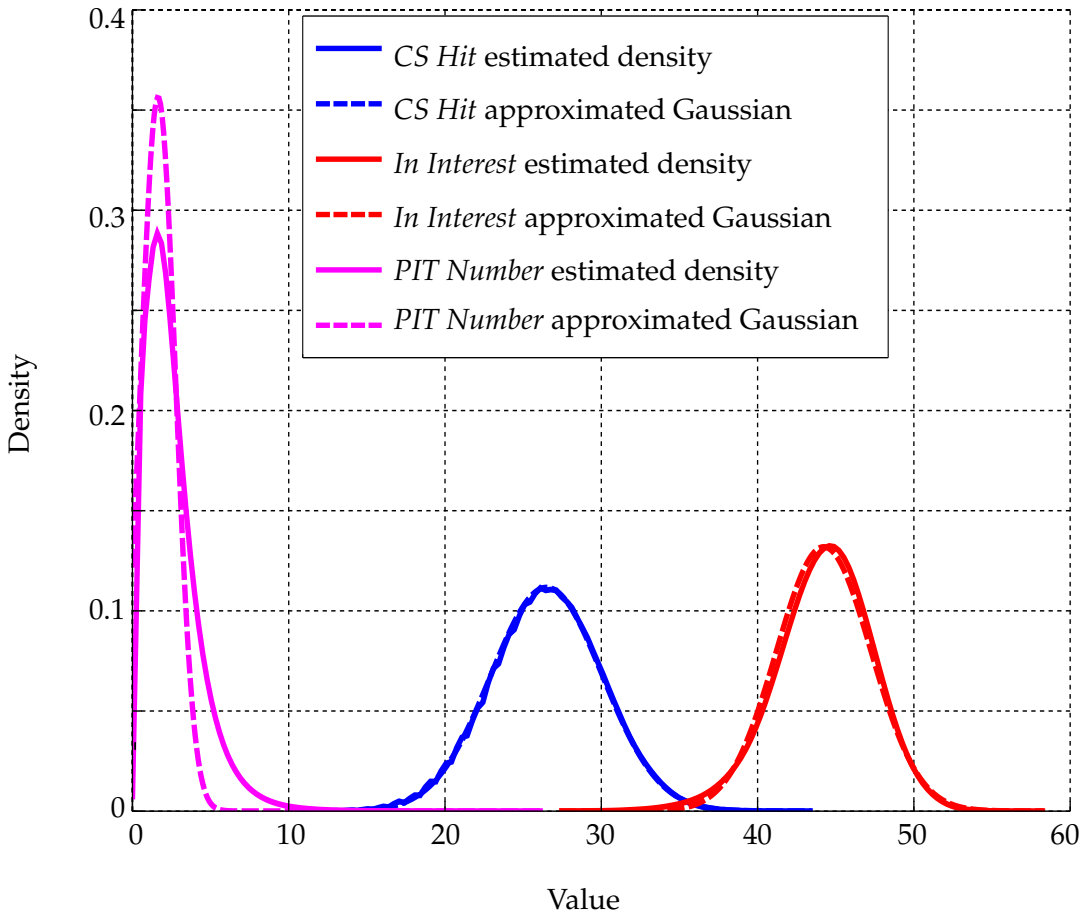


FIGURE 3.9: The kernel estimated density function and the normal distributions of illustrative metrics

guarantee the prescribed PFA of the micro detector and the relevance of the model. Meanwhile, for a few metrics (e.g. *PIT Number*), our micro detector cannot ensure the performance for small prescribed PFA. As stated in the previous subsection, this phenomenon is due to the fact that these metrics are not well modeled by the normal distribution. However, as shown in the following section, a performance enhancement is possibly obtained by combining micro detectors, hence the modeling errors on their distribution is compensated by each other.

Learning Parameters of Proposed Bayesian Network

To evaluate the learning efficiency of BNC when the size of training set varies, we use the usual k-folds cross-validation method with $k = 5$. Consequently, the dataset is divided into 5 subsets. Each subset will, in turn, be used as testing data and the remaining four will be used as training data. The average misclassification rate (i.e. the total number of misclassified samples over the total number of samples) over training subsets is defined as *train error*, while the one obtained over testing subsets is called *cross-validation error*. Figure 3.11 shows the learning curves of the proposed BNC when the size of training dataset per scenario changes. When the size of training set per scenario increases, the misclassification error starts decreasing. An

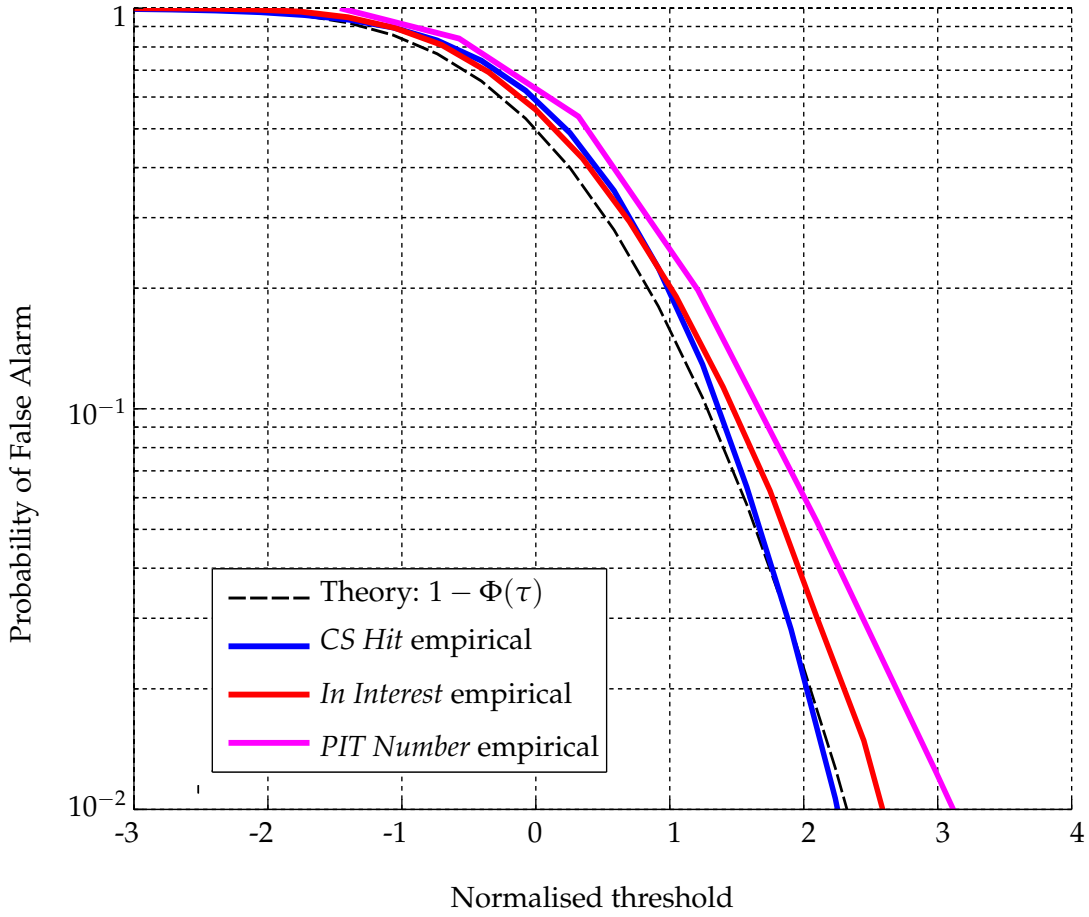


FIGURE 3.10: Guarantee of prescribed probability of false alarm for micro-detectors of illustrative metrics

optimal value is achieved around 280 training samples per scenario. After that, the misclassification error keeps increasing due to the well-known phenomenon of overfitting. Therefore, the optimal value of 280 samples for the training set per scenario has been chosen, corresponding to about 23 minutes of samples collection.

Bayesian Network Classifier Evaluation

Figure 3.13 and 3.14 depict, respectively the accuracy (i.e. total number of samples classified correctly over the total number of samples, or one minus the misclassification error) and the delay of the proposed BNC when the attacker rate changes.

Regarding the attack rate's impact, the figures indicate that when the attack rate is less than the *Interest* rate under normal traffic, BNC has a low accuracy, a high delay and those results have a large statistical spread. On the other hand, BNC achieves over 95% of accuracy with a delay of about one sample when the attack rate is higher than normal user rate.

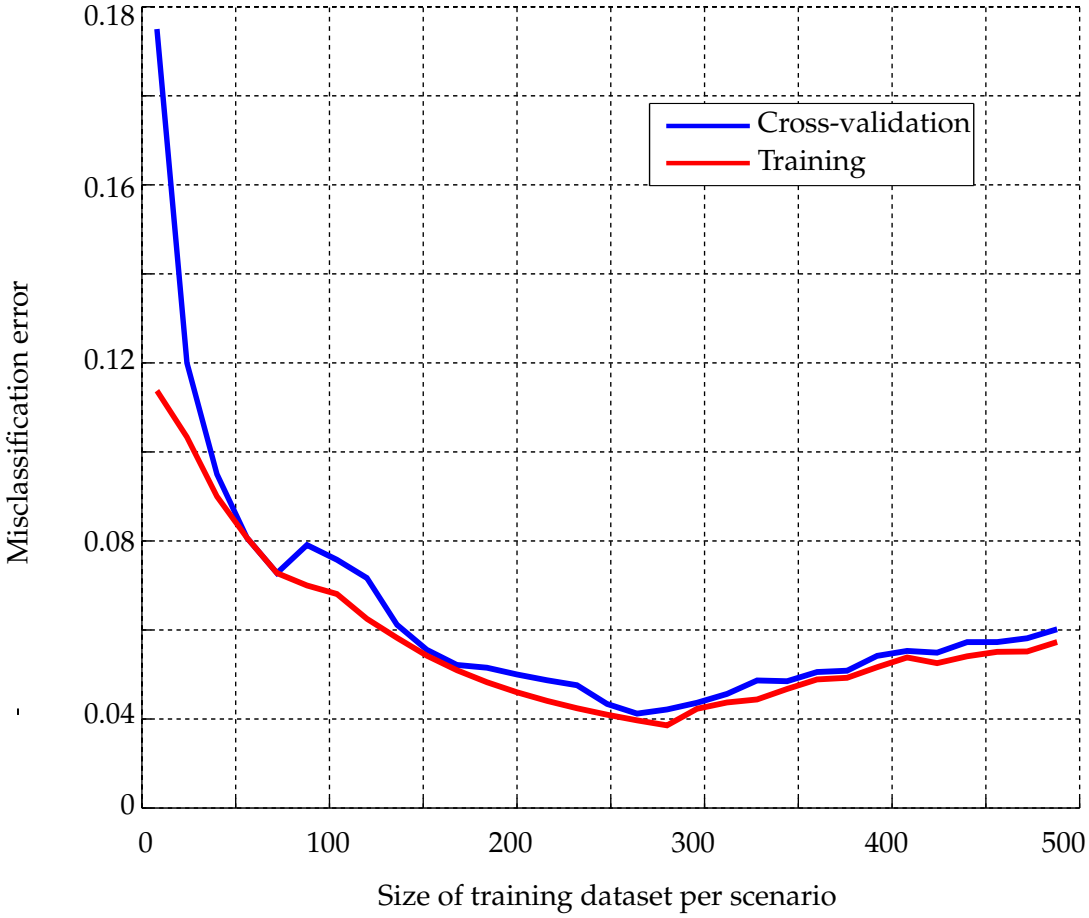


FIGURE 3.11: Learning curve of the proposed Bayesian Network

With regard to the impact of attack scenario, one can note from Figure 3.13 and 3.14 that the accuracy and delay of BNC against *CPA multicast* is better than in *CPA bestroute*. The reason is that, in *CPA multicast*, the *Interests* are more likely to be forwarded to the bad provider without much effort from the attacker, so the behavior of attack in any rates is visible.

Concerning to the location's impact, in *CPA bestroute*, BNC at R1 is more accurate than in R2 router because it receives *Interest* packets from clients and attackers first. As a result, its metrics will be more affected than those of R2. Meanwhile in *CPA multicast*, since R2 is more likely to be poisoned due to the delay advantage of the bad provider, its metrics are impacted more obviously than ones of R1. Therefore, BNC in R2 achieves a better accuracy than in R1.

Eventually, it is important to point out that the accuracy of BNC can be improved by increasing the detection window n of the micro detectors. As shown in Figure 3.12, for the worst case of *CPA bestroute* with 1 Interest/s, the accuracy increased from 53% up to 93% by raising the n from 1 to 5. Nevertheless, it is worth noting that increasing the detection window will increase the detection delay of micro detectors. This trade-off should be considered carefully in the deployment.

When the attacker has the same *Interest* rate as that of a good client, the accuracy

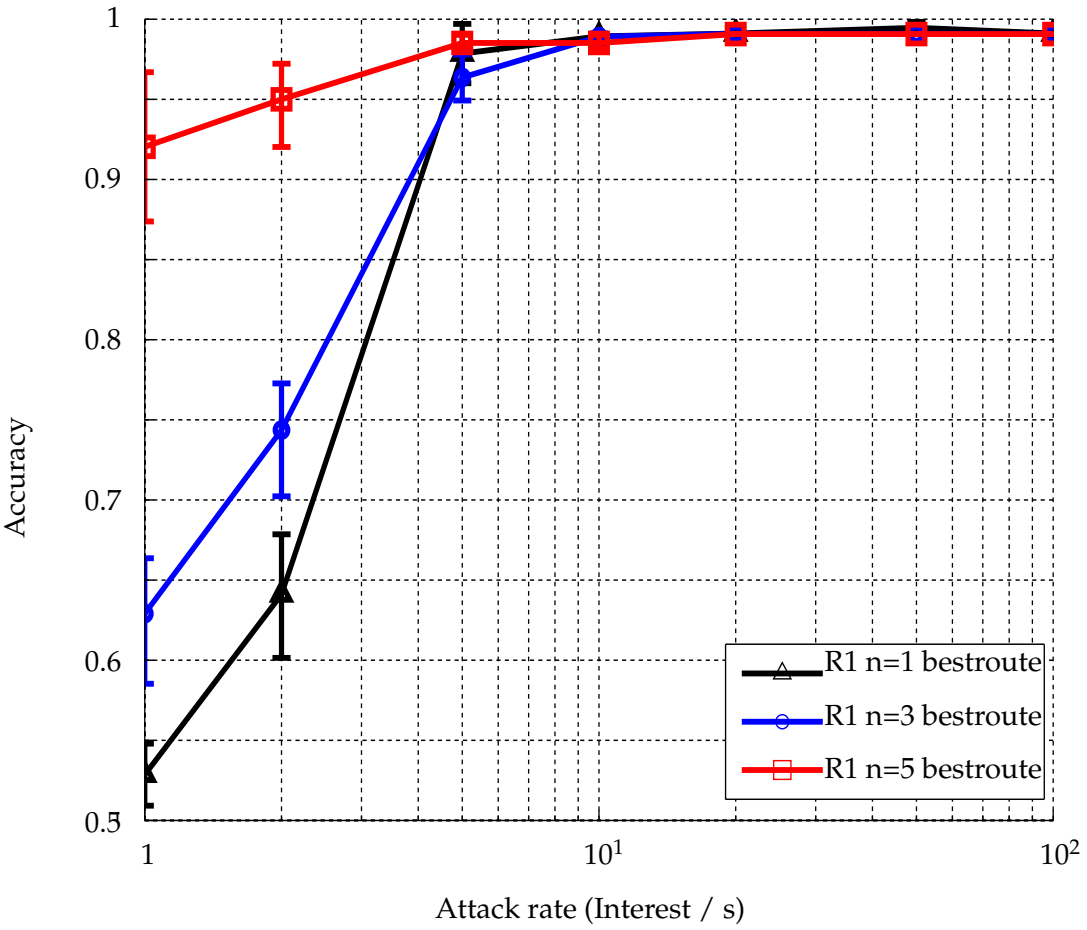


FIGURE 3.12: Performance of the proposed classifier with different detection window

of the proposed detector is higher than 0.95. Even when the attacker's *Interest* rate is very low as compared to the *Interest* rate of the good client (1/10), the accuracy is still higher than 0.7. As a consequence, we can conclude that our local detector is reliable even in a small-scale attack.

3.6 Conclusion

In this chapter, we presented a complete security monitoring plane for NDN. This plane is built on a three layer architecture offering a fully distributed solution combining advanced probes and micro detectors distributed in the NDN routing hosts. To be able to detect attacks, we built a collection of 18 NFD metrics based on a thorough analysis of the NFD pipelines. For each metric, a micro detector has been designed to find any irregular variation from the normal behavior at the prescribed false-alarm rate. The applicability and performance of our micro detector solution was successfully assessed in a testbed against CPA.

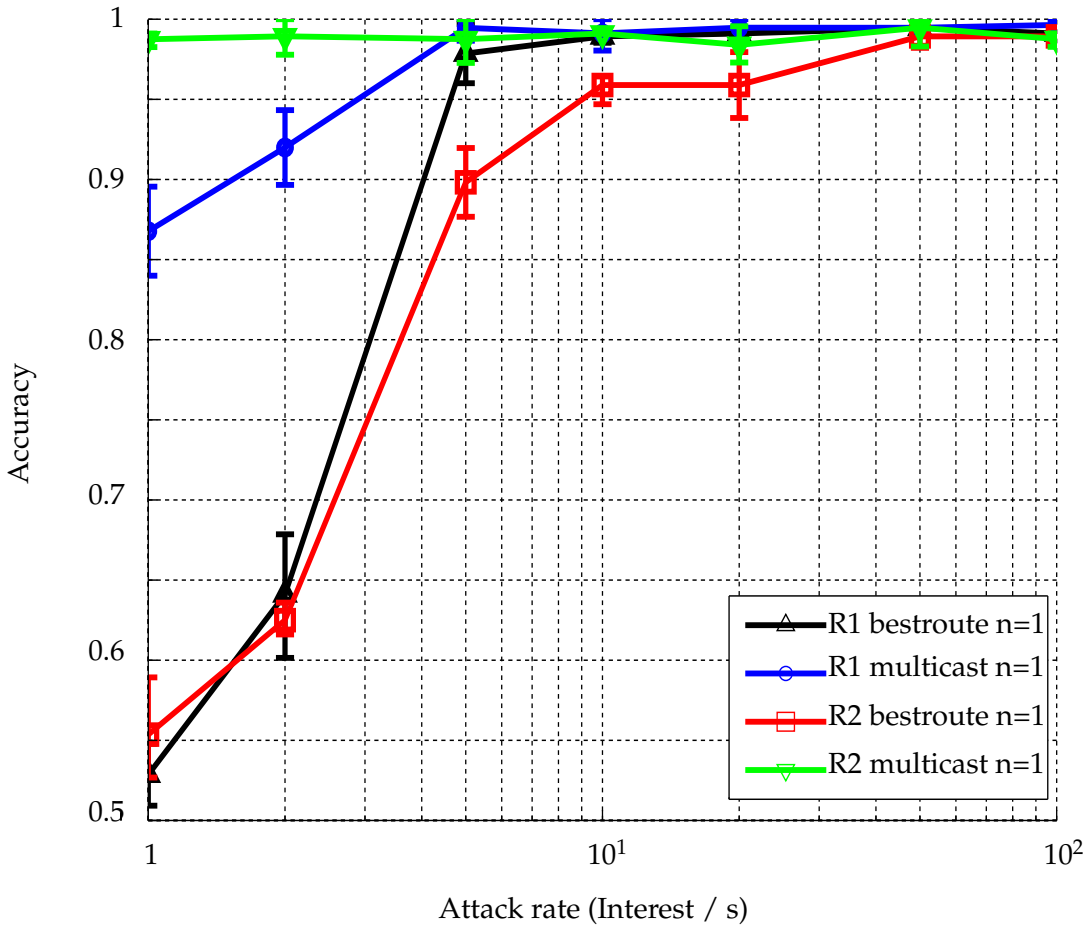


FIGURE 3.13: Accuracy of the proposed classifier with different attack rates

For attacks involving many facets of NDN node status, such as CPA, we designed a Bayesian Network-based Correlation Engine to correlate micro detector alerts to recognize any suspicious security events in the NDN node. To validate the entire architecture in an integrated way, two attack scenarios of the CPA have been considered in a real testbed implementing all the contributions. This integration and extensive testing demonstrate the capability of our solution to detect these attacks with various rates accurately.

The monitoring plane is the first major element of an entire autonomic security plane. To close the loop, a remediation deployment approach to effectively stop the propagation of attacks is needed. To deal with scale, we also need to offer ways to distributed the workload of our anomaly detectors while reducing the delay and resource consumption. These issues are addressed in the following chapters.

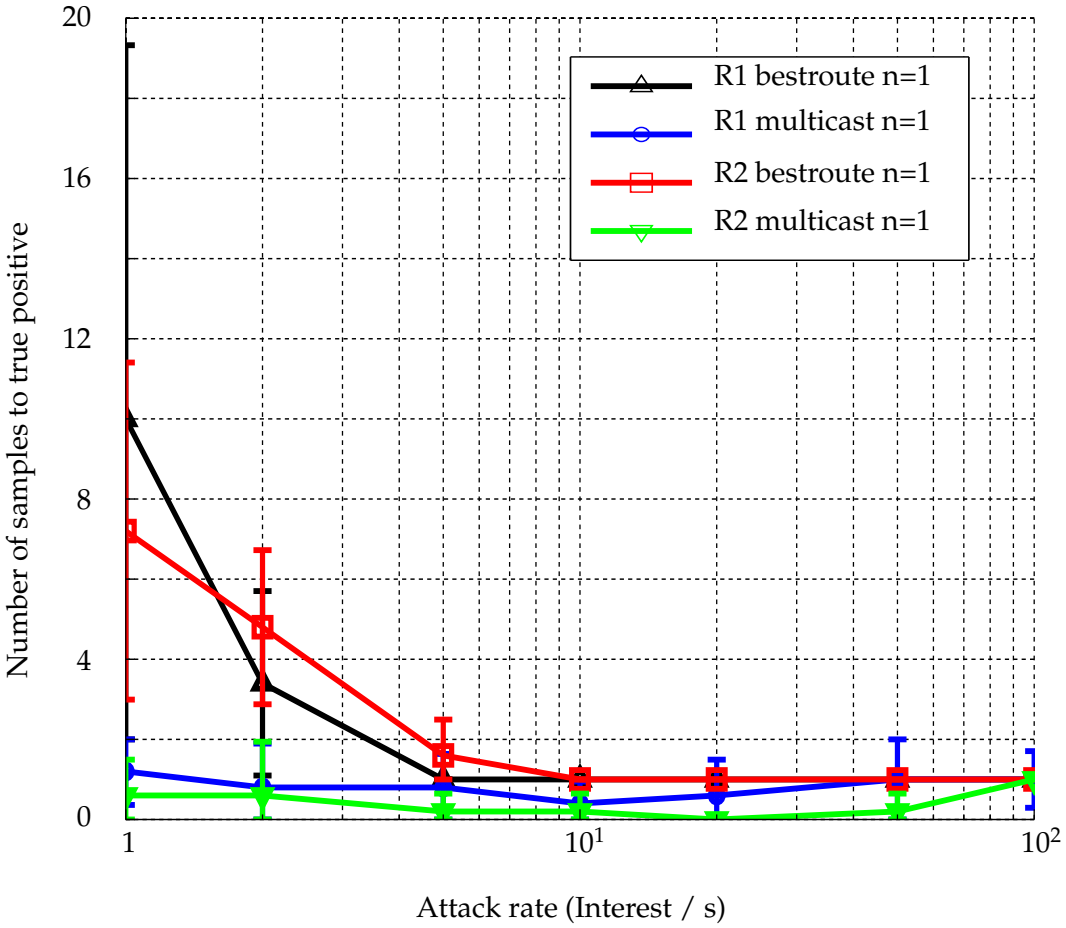


FIGURE 3.14: Delay to detect attack with different attack rates

Chapter 4

Toward a Secure Content-Oriented Orchestration

Contents

Even if filled with promising features and great application targets, no novel network architecture can be deployed at large scale without strong secure and efficient underlying network and service management support. While NDN has the promising features in terms of impact, it still misses a solid management framework. The proposed monitoring plane we designed and described in chapter 3 enables monitoring and detection of both anomalies and security incidents in NDN

networks and can serve as the data collection and analysis of a management plane. To close the loop, we need to build the action part of the plane to host the countermeasures against observed incidents. To this end, we propose in this chapter a Secure Content-Oriented Orchestration built on standardized architectures and well established design principles. We argue that the Network Virtualization Function (NFV) design, which allows network operators to deploy network functions in software running on standard commodity server hardware coupled with SDN, is an excellent candidate to boost ICN deployment. To support our approach, we design and implement a novel NFV MANO architecture to deploy, manage, monitor, and secure any NDN virtual network. First and foremost, we present the motivation for our secure content-oriented orchestration. Next, the three main contributions are presented: (1) the extension of the TOSCA profile for NDN, (2) the modification of the profiles for content-aware components, and (3) a fully operational content-oriented orchestration. These contributions have been integrated in a common framework, deployed and tested. The results obtained from the different experiments we performed to evaluate the performance of our content-oriented orchestration are analyzed in the chapter together with lessons learned. The work presented in this chapter was conducted in collaboration with Messaoud AOUADJ - R&D Engineer at UTT. The contribution in this chapter has been presented in [121].

4.1 Motivation for a Secure Content-Oriented Orchestration

NFV MANO functional blocks are standardized by the ETSI organisation and implemented in current NFV frameworks [33]. Existing instances are only able to host IP-based Virtual Network Functions (VNFs). NDN's features are not recognized and thus, not yet supported by this well established framework. For example, MANO has features to deal with IP-based routing and addressing but since these functions are fundamentally different in NDN, the management models do not match. NDN also introduces new entities such as NDN firewalls, gateways which are unrecognizable in the existing MANO service model. The simplest operation of NDN virtual network is therefore not feasible in the current frameworks.

The overcome the missing capacities of the current MANO solutions and the standard NFV framework, we design a particular orchestration for NDN, which employs the NDN Content-Oriented approach. Among the different frameworks, as explained in section 1.2.3, the DOCTOR MANO architecture, is the best choice for an NDN orchestration solution. It is introduced in the next section followed by a detailed presentation of the extensions we provide to enable orchestration, also described and assessed in this chapter.

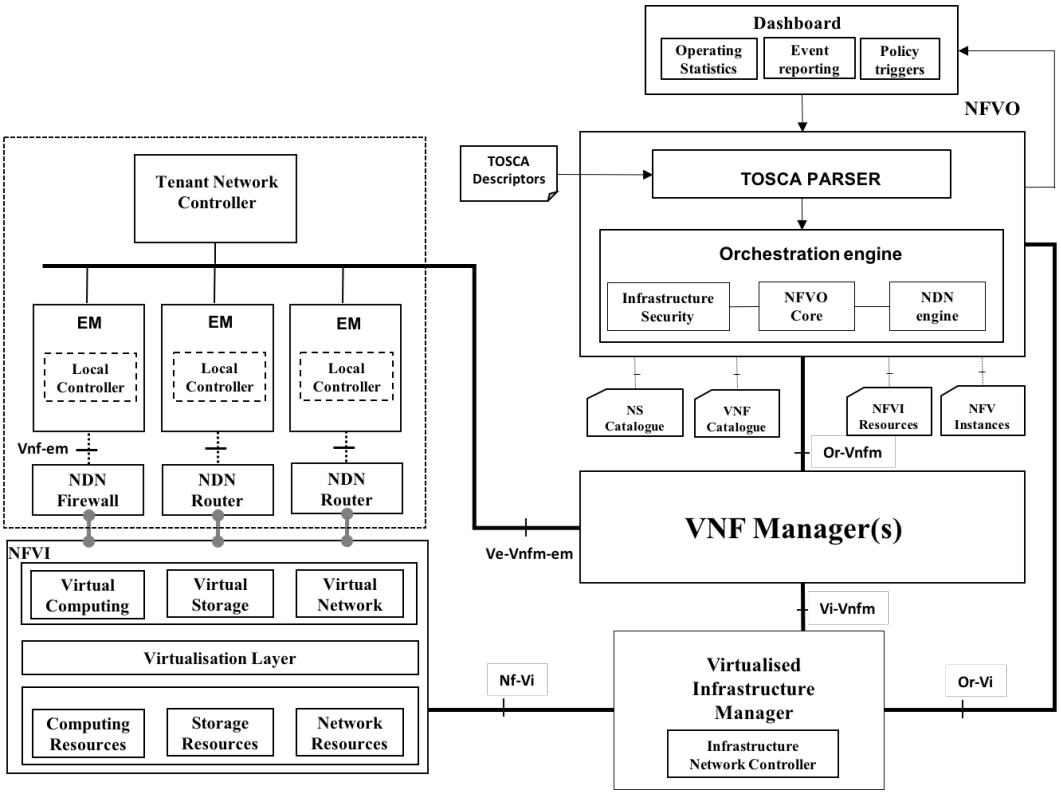


FIGURE 4.1: DOCTOR Virtualized Network Architecture

4.2 DOCTOR's NFV MANO Architecture

The DOCTOR Virtualized Network Architecture, which is illustrated in Figure 4.1, rigorously follows the ETSI reference architecture specification [33]. To support ICN oriented operations, its MANO components were refined and instanciated. These are :

- Network Functions Virtualization Infrastructure (NFVI): The NFVI includes all network, computing and storage resources (hardware and software) needed to deploy and connect Virtual Network Function (VNF) in carrier networks.
- Virtual Infrastructure Manager (VIM): DOCTOR's VIM is the piece of software (Docker through its Swarm API) responsible for controlling and managing the NFVI compute, storage, and network resources.
- Virtual Network Function: DOCTOR's VNFs are dockerized applications. They include: the DOCTOR ingress and egress gateways, the NDN router, the signature verification module and an NDN-based firewall.
- Element Manager (EM): The EM is responsible for the FCAPS management functionality associated to a VNF. This EM includes a local controller unit

(MMT Local Controller) that provides monitoring, management, and control of a virtualized NDN function.

- MMT Tenant Controller: The tenant controller is the strategic control point for the NDN virtualized network. It manages and controls the virtualized underlying network according to NFVO requirements. Following the SDN logic, the MMT Tenant Controller represents a central point of control and maintains a global view over the deployed virtual network.
- VNF Manager (VNFM): The VNF Manager is responsible for the lifecycle management of VNF instances. In the context of the DOCTOR architecture, each network tenant has its own VNFM.
- TOSCA processor: TOSCA, which is an OASIS standardized language, is the de-facto standard for modelling Cloud and NFV applications. The TOSCA processor reads the TOSCA templates and creates an in-memory graph of TOSCA nodes and their relationship. The in-memory graph is then passed to the NFVO engine which extracts deployment and service functions chaining information.
- NFVO core represents the glue between all others NFVO modules. In particular, NFVO core includes all the processes that allow transforming a TOSCA specification to a set of instruction to send to other MANO blocks.
- NDN engine centralizes the NDN knowledge of the orchestrator. This generally represents the routines and instructions that allow the orchestrator to deal with ICN key concepts, such as *Interest* and *Data* packets.
- MMT infrastructure security ensures multi-tenancy security. The module collects operational information and security reports from all VNFMs. if the analysis reveals an attack or vulnerability exploit, a report is sent to the administrator (through the dashboard) so that she can trigger appropriate remediation, whether automatic (already on-boarded TOSCA templates) or manual.
- MMT Dashboard: MMT dashboard is a human-friendly interface to the NFVO, used mainly for visual monitoring by administrators.

4.3 Extended TOSCA profile for NDN networks

Like any orchestration network architecture, NDN needs support for a topolocial description. As such, templates are needed for *Virtual Deployment Unit (VDU), VNF, Virtual Link (VL), Connection Point (CP)* and *Forwarding Path* description. Several languages such as TOSCA and YANG have been carefully considered in chapter 1. We

selected TOSCA since it effortlessly defines the deployment, configuration and the life-cycle management of a service thanks to its concepts. Moreover, TOSCA is easy to extend and well-adopted as a Domain-Specific Language in the NFV industrial and academic worlds. We extended the TOSCA profiles and policies to fulfil two goals of a content-oriented orchestration : (1) deployment automation and (2) remediation deployment.

4.3.1 NDN TOSCA base nodes

To address the lack of NDN knowledge in current MANO architectures, specific TOSCA [126] profiles are extended to include advanced NDN features. To enable the capture of the concept and capabilities of TOSCA language, let's go through a simple TOSCA template deploying a single server.

```
tosca_definitions_version: tosca_simple_yaml_1_0
description: Template for deploying a single server.
topology_template:
  node_templates:
    my_server:
      type: tosca.nodes.Compute
      capabilities:
        host:
         properties:
           num_cpus: 1
           disk_size: 10 GB
           mem_size: 4096 MB
        os:
          properties:
            architecture: x86_64
            type: linux
            distribution: rhel
            version: 6.5
```

The above TOSCA model describes a Compute node with its associated properties. All orchestrators able to read the TOSCA profile are supposed to instantiate the defined Compute node fulfilling its capabilities. More specifically, the "host" capacity allows administrators to specify the number of processors, memory size, and disk size required by any instance of the node. Likewise, the "os" capability is used to provide values indicating which host operating system the compute node must run.

The MANO framework already has a couple of node specifications related to network topology: *VDU*, *VL*, *CP*, and *ForwardingPath*. For NDN deployment, the *VDU, VL* and *CP* do not need to be extended and can be reused without any modification, as they relate only to the infrastructure layer which is technology agnostic. Thus, only the forwarding element needs to be redesigned for virtualized NDN networks. Also, the TOSCA nodes for VNF, *Forwarding Path* are extended since NDN uses announced name prefixes and content information together with demand information rather than the generic L3 and L2 source and destination addresses to forward both data and signalling in the network. To make this data visible, our *VNF* node includes configuration parameters that represent the set of announced NDN prefixes.

To manage security, a signature verification module is introduced in each node. By default, this module is not activated because of both high resources consumption and delay added by the signature verification process. This process can be activated on demand and its status is included in the VNF specification.

The *Forwarding Path* specification has also been extended to catch the list of VNFs that will process a specific set of NDN packets. As the NDN router is not aware of the final destination of each request, our *Forwarding Path* specification is specified by either the forwarder or the NDN prefix that is added as the *Forwarding Path policy*.

The following TOSCA template contains the TOSCA grammar for the *Forwarding Path* in an NDN router:

```
tosca.nodes.nfv.doctor.FP:
  derived_from: tosca.nodes.Root
  properties:
    id:
      type: integer
      required: false
    policy:
      type: tosca.nfv.datatypes.policyType
      required: true
      description: policy to use to match traffic for this FP
    path:
      type: list
      required: true
      entry_schema:
        type: tosca.nfv.datatypes.pathType
```

In the TOSCA grammar above, the two essential capacities to enable NDN is the policy and path, which are derived from *tosca.nfv.datatypes.policyType* and *tosca.nfv.-datatypes.pathType* respectively. The policy capacity contains the NDN name prefix as illustrated in the TOSCA template *tosca.nfv.datatypes.policyType* as follow:

```
tosca.nfv.datatypes.policyType:
      properties:
        type:
          type: string
          required: false
          constraints:
            - valid_values: [ NDN ]
        prefix:
          type: list
          required: true
          entry_schema:
            type: string
```

The path capacity contains a list of policy types (*tosca.nfv.datatypes.policyType*). Each policy Type is an interface of the NDN router, more specifically, it contains the forwarder and capability which are the name of the node and the connection point respectively:

```
tosca.nfv.datatypes.pathType:
  properties:
    forwarder:
      type: string
      required: true
    capability:
      type: string
      required: true
```

The extension of NDN TOSCA base nodes provides the initial pieces to feed a content-oriented orchestrator. To support these extensions in the TOSCA profiles, existing orchestrators need to be extended. Besides the core components TOSCA profiles presented so far, TOSCA policies need to be extended as well so as to address the remediation deployment. We will present this modification in the next section.

4.3.2 NDN TOSCA policies

The goal of our NDN TOSCA is to guide the system when a performance or security incident occurs. Theses policies are dynamically activated during service runtime in

order to help the system to react to pre-defined performance or security events. To automate the reaction, the TOSCA policies modeled with Event-Condition-Action (ECA) rules are selected. The NDN TOSCA policy model contains a list of targets subject to the policy together with a list of triggers. Each trigger contains the triggering event, the condition used to enable the policy and the mitigating action. The following TOSCA template contains the grammar of an NDN TOSCA policy:

```
- <policy_name>:
    derived_from: tosca.policies.Root
    description: <policy_description>
    type: <policy_type>
    targets: [<list_of_targets>]
    triggers:
      <trigger_name>:
        event_type: <event_type>
        condition:
          constraint: <...>
        action:
          action_type: <action_type>
```

Two types of policies were defined in our work : one that deals with dynamic reconfiguration of functions and one performing scale-out operations. The reconfiguration policy allows to dynamically change the state of an NDN node, such as the enforcement of signature verification to be applied on *Data* packets, which is called the signature verification policy. Each policy includes the identifiers of the target VNF as policy enforcement points and the event which triggers this action, which is, in the case of security policies, an alert issued by a detection engine. As a second example of a dynamic reconfiguration policy, one can consider the update of the white and blacklist of an NDN firewall. In a way similar to the signature verification policy, it specifies the target firewall and the content prefix specified in an event which induces the configuration update. Finally, we have defined a scale-out policy which allows any NDN routing component to be dynamically replicated as soon as the PIT size of a given NDN router crosses a given threshold. This generates a dedicated event which triggers the countermeasure. Indeed, the PIT size is a good indicator of the routing load of a node since it stands for the number of *Interests* that have been forwarding without having received a *Data* packet in return; as such it stands for the stateful component of an NDN router that can lead to resource exhaustion.

The extension of TOSCA policies follows a *When*, *Which* and *What* specification. *When* stands for the moment to trigger the action, it could be the detection of an attack or an excess of a metric. *Which* stands for the corresponded nodes that

need to be updated. *What* stands for the precise measures that need to be performed in corresponding nodes. By answering these questions, the TOSCA policies help *content-aware components* to perform the necessary dynamic configuration to react against attack or performance incidents. These *content-aware components* are developed in the next section.

4.4 Content-aware components

Our approach consists in remaining as compliant to MANO as possible and therefore solely expanding or redesigning the components of MANO that involve NDN knowledge, without diverting them from their original purpose. These are the *VNF Manager (VNFM)* and the *NFV Orchestrator (NFVO)*. They are extensively described in the context of the deployment and management of an NDN virtual network.

4.4.1 NDN VNFs

NDN router and NDN firewall are the two first NDN VNFs in our architecture. They are both dockerized applications.

The NDN router itself is an extension of the official NDN distribution. The extensions focus on security and include our monitoring probe and the signature verification module introduced above. The probe is used to collect the different NDN metrics and correlate them to identify anomalies or potential attacks [127]. The signature verification module is the one proposed in [35]. Each time the module receives a *Data* packet, it checks if it has a known *KeyLocator* and in that case, it performs a signature verification. Depending on the content management rules and the verification result, the module either drops the packet or forwards it to subsequent nodes.

The standard NDN Forwarding Daemon (NFD) does not support dynamic appending of filtering rules to perform *Interest* packet filtering (e.g. spam filter). Although *NFD* could be modified to install filtering functions, that may cause significant performance degradations. Besides, this function is not mandatory in all routers of a topology (usually only relevant in edge nodes). As a consequence, we considered a standalone NDN firewall to be a VNF whose description can be found in [128]. This firewall can be easily configured by our orchestration solution to add or remove filtering rules.

4.4.2 NFVO

The *NFVO* core functionalities need to be updated in order to take into account the characteristics of the NDN paradigm. These features include the use of prefixes instead of traditional L2 and L3 addresses for flow classification, as well as internal information such as *FIB*, *PIT* and the embedded cache in each NDN router. To integrate them, we have designed and implemented a dedicated *NFVO* which includes two main blocks: a *TOSCA parser* and an *Orchestration engine*. The *TOSCA parser* reads TOSCA templates and builds an in-memory graph of the TOSCA nodes and their relationship (step 1 & 2 - Figure 4.2). The graph is then transferred to the *Orchestration engine*. In the *Orchestration engine*, the *NDN engine* centralizes the NDN knowledge of the *NFVO*, which generally represents the routines and instructions that allow the *NFVO* to deal with NDN key concepts, such as *Interest* and *Data* packets. More precisely, when the in-memory graph of the TOSCA nodes is obtained, the *NDN engine* extracts the forwarding information specified in the *forwarding path* from the in-memory graph (step 3) and translates it into the NDN routing configuration for each VNF (step 4). This includes the NDN prefix, the address of the next-hop and the port number of the NDN application (e.g. *NFD*, NDN firewall or signature verification module). In addition to the initial configuration defined in the deployment specification of the NDN TOSCA profile, the *NFVO* also considers the actions to be performed under the NDN TOSCA policy. The NDN routing specification may, therefore, include other details, such as an enforcement action or a forwarding strategy for the NDN prefix. The acts that may be taken are: insert, modify and delete.

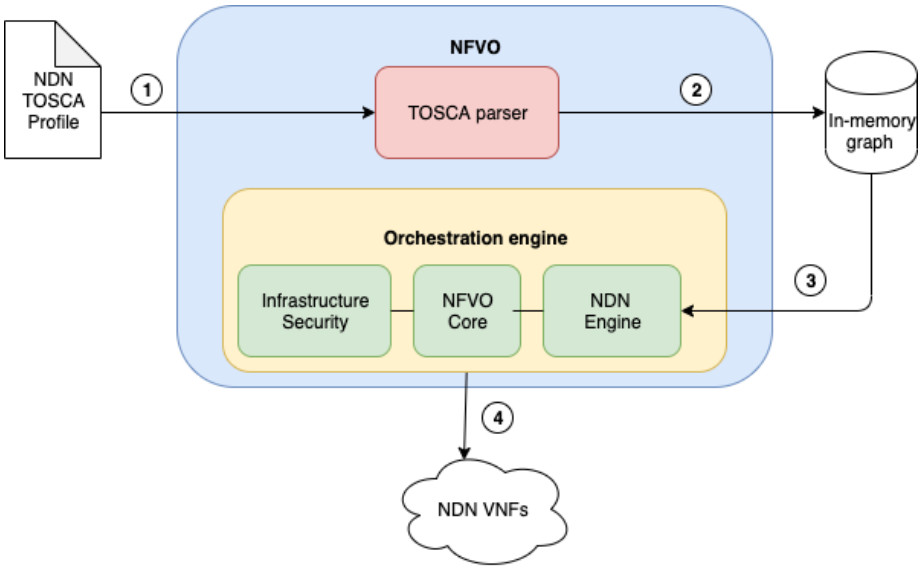


FIGURE 4.2: The structure of NFVO

4.4.3 VNF Configuration Management

The *VNFM* is responsible for the life-cycle management of VNF instances. In the context of NDN, the *VNFM* is extended to relay all specific NDN configurations (e.g. the NDN prefix-based routing information, the NDN prefix to be blocked in NDN firewall, etc.) issued by the orchestrator to all target *NDN VNFs*. It is also extended to receive notifications from the different *Element Managers (EM)*[1].

The Element Manager leverages the NFD management protocol[2] to configure an NDN router. Specifically, the *EM* retrieves the list of *faces*, the list of entries in the *FIB* and the forwarding strategy via the *Status Dataset* of the NFD management protocol. These lists are compared to the configuration information received from the VNFM that needs to be applied and, depending on the action to be taken (add/update/remove), it performs the necessary modifications thanks to the *Control Command* services of the NFD management protocol. The *EM* also includes a local controller unit (*Local Controller*) which represents a local point of control and embeds security applications that allow it to detect NDN attacks locally. In case of an attack, it issues notifications towards the *VNFM*.

4.5 Content-Oriented Orchestration

Given the static description of ICN-aware MANO components provided above, we now explain the two main orchestration processes that are performed by the *NFVO*, deployment automation of an NDN virtual network and dynamic enforcement of policies.

4.5.1 Deployment automation

Figure 4.3 depicts the general scenario for a virtual NDN island deployment. Firstly, the *VIM* deploys a management network to ensure the communication between VNFs and the *VNFM*, followed by the deployment of the *VNFM*'s container connecting to this network. Once the *VNFM* is deployed, it receives the different configurations from the *NFVO* to deploy virtual links (*VLs*) and VNFs and it forwards them to the *EM*. Using the NFD management protocol, each *EM* retrieves the list of faces in NFD and verifies if its related NFD already has a face to the next-hop, if not it creates a new one. Next, each *EM* updates the routes for prefixes in the *FIB* by taking care of potential already existing entries. Each *EM* also considers the forwarding

[1] EM are internal components responsible for all management purposes in a *NDN VNF* - See section 4.2

[2] https://redmine.named-data.net/projects/nfd/wiki/Management

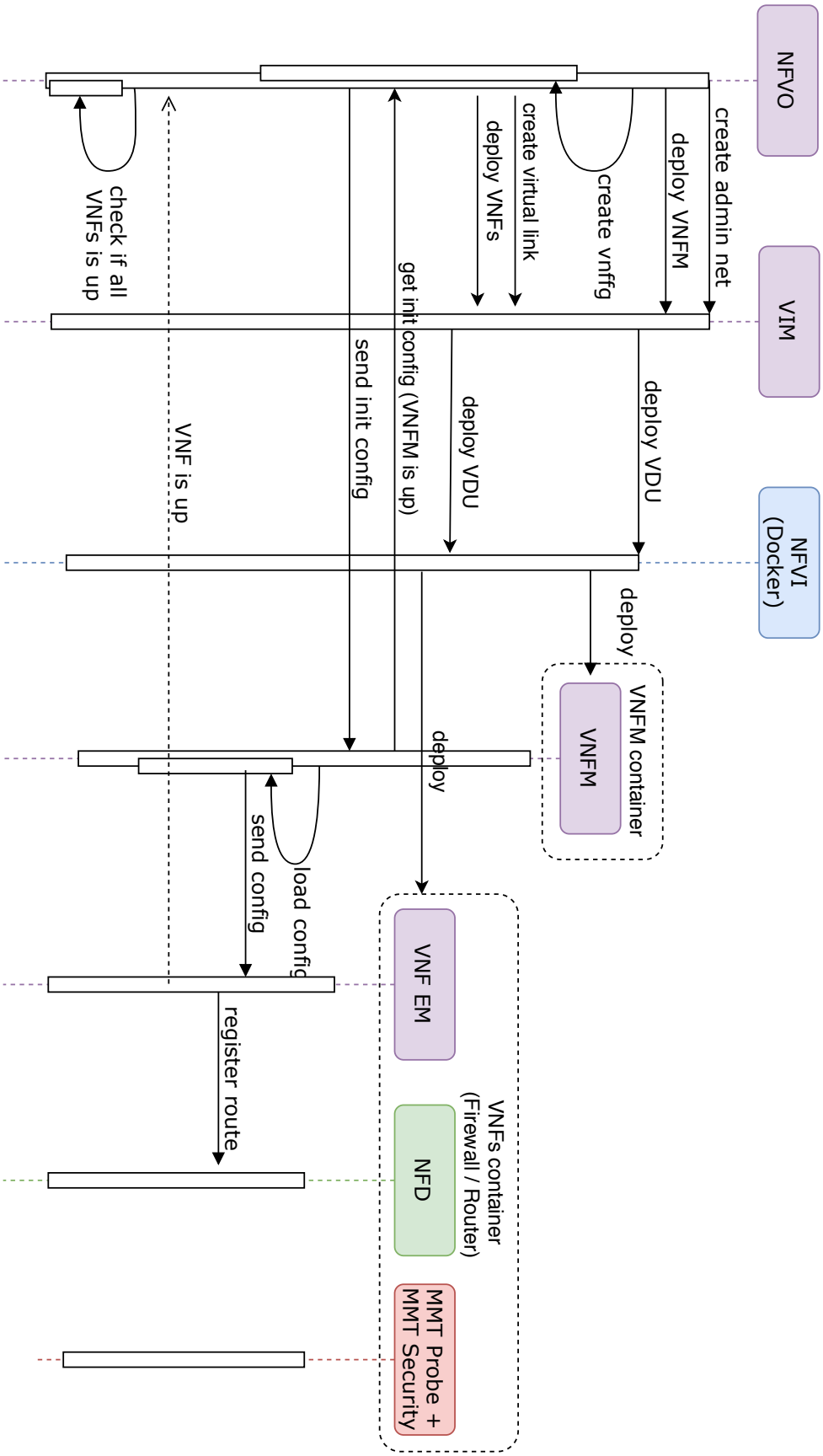


FIGURE 4.3: Sequence Diagram for the deployment

strategy of NDN for the different prefixes such as *best-route, multicast or round-robin*, which can be configured using the *Strategy Choice* of the NFD management protocol. To maintain the synchronisation of configurations, each *EM* periodically verifies the *FIB* of its related NDN router to ensure that the *NFD* process operates the configuration received from the *VNFM*. the EM can trigger a reconfiguration if needed. As such, each *EM* operates as a self-healing configuration service.

4.5.2 Dynamic configuration of functions

Triggered by an event defined in the TOSCA policies, the *NFVO* can also dynamically reconfigure any virtualized NDN function with a deployment-like process. The different reconfigurations include all updates of the prefixes to block in an NDN firewall, the activation of a signature verification module and the update of the NDN routing configuration in NDN routers. All reconfiguration operations are performed under the control of both the *NFVO* and *VNFM*. In the context of NDN, these reconfigurations are more distinctive as each configuration can affect many nodes. The route for the NDN prefixes are registered in each router to which the set of NDN packets will be forwarded to. Consequently, if other routers do not become aware of the modification, the request may not be forwarded to the correct node. Therefore, the *NFVO* must integrate the change and update both its topology and graph. Then, it has to perform the modifications in affected nodes and finally announce the modifications to *VNFs* via the *VNFM*. The whole process can be executed thanks to the *NDN engine* which is integrated into the NFVO which contains the routines and instructions that allow the *NFVO* to deal with NDN key concepts. The following sections will explain in detail how our proposed *MANO* embraces the dynamic configuration of functions to perform the remediation deployment.

4.5.3 Attack mitigation deployment

Attack mitigation deployment for NDN uses the extension of NDN TOSCA policies and the built-in dynamic configuration of functions. Based on the policies defined in the TOSCA profile, the *NFVO* chooses the corresponding measures to counter against attacks. In the case of the Content Poisoning Attack for example, the countermeasure that is defined in the TOSCA policies is the activation of the signature verification module. This module allows the system to detect poisoned content, add the corresponding names to the blacklist of a NDN firewall, which will in turn, blocks the bad content. The execution of these two policies is based on the dynamic configuration of functions, which allows in the first step to activate the signature verification

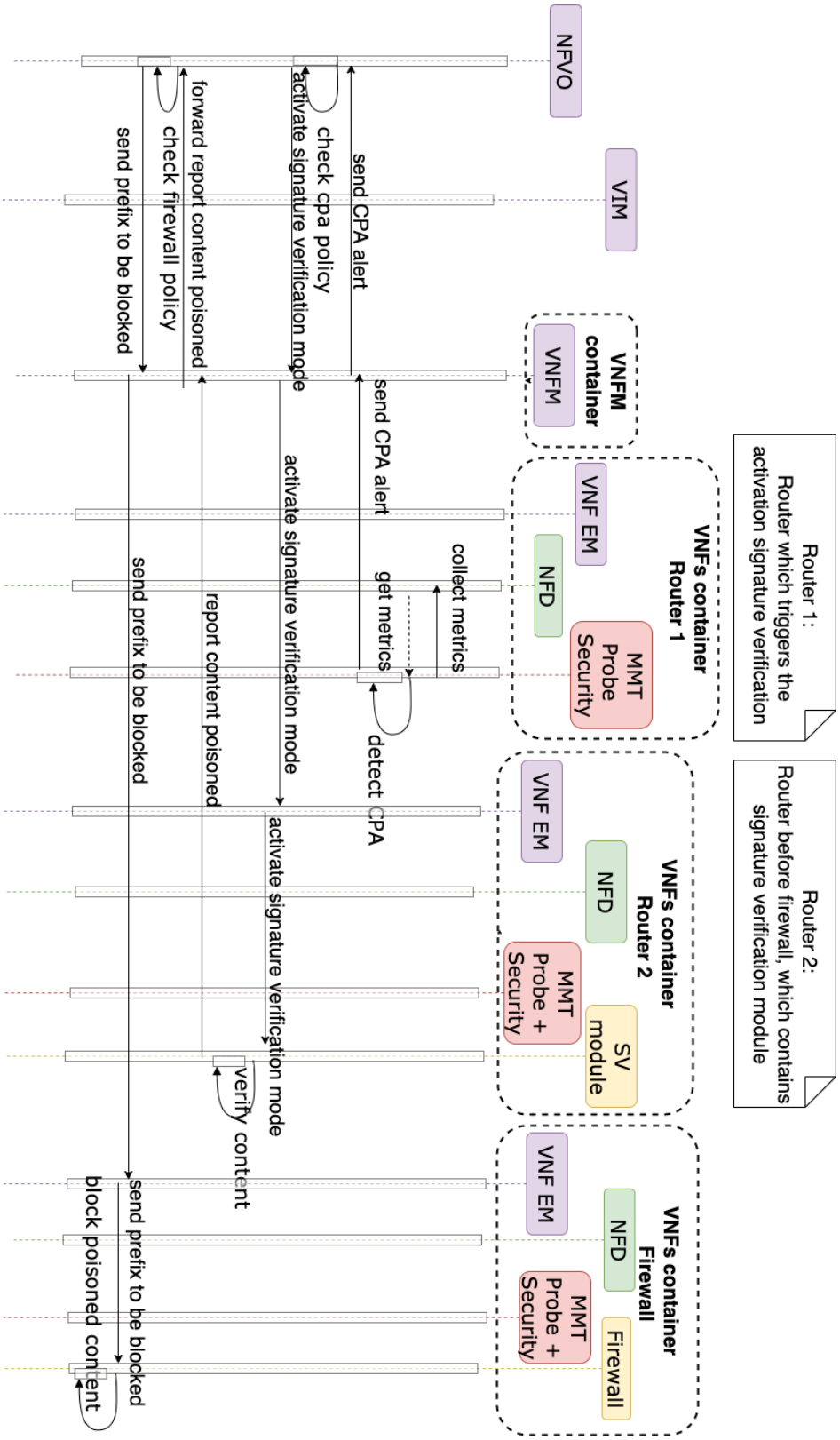


FIGURE 4.4: Sequence Diagram for the CPA Detection policy triggered

module in several nodes, then in the second step to add poisoned prefixes into the blacklist of the firewall.

Figure 4.4 illustrates such a process. To facilitate the reading, we named *Router 1* as the router which detects the attack and triggers the activation of the module signature verification in the egress router and *Router 2* as the ingress router hosting the firewall, which also contains module signature verification.

Whenever *Router 1* detects the attack, it sends an alert to the *NFVO* via *VNFM*. In turn, *NFVO* stores the NDN knowledge thanks to TOSCA parser and NDN engine. Thus *NFVO* can recognize the attack and triggers the counter-measures against an attack based on the policies defined in the TOSCA profile if the conditions are matched. Several counter-measures can be used to remediate the CPA; in the context of our study, we propose to use a firewall coupled with the signature verification module to block the malicious content. This choice is defined in TOSCA policies which instruct *NFVO* to send a dynamic configuration to *Router 2* which demands this router to activate its signature verification module. From this moment, content poisoned will be detected and sent to the *NFVO* via *VNFM*. Once again, whenever *NFVO* receives a report of content poisoned from a *VNF*, it checks the policies defined in the TOSCA profile. The action corresponding to this event is the blockage by the firewall of poisoned content. The actions will also be done by the dynamic configuration from *NFVO* to *VNFM* then *Routers*.

As shown above, the remediation deployment is guaranteed in our proposed orchestrator thanks to the dynamic configuration of functions and predefined-policies. Besides, the dynamic configuration also allows to remediate performance incidents, which will be explained in detail in the next section using a standardized measure for performance incidents.

4.5.4 Scale-out

Scaling-out a service is an essential feature for a MANO architecture to guarantee the performance and network throughput. A scale-out operation must allow dynamic provisioning of additional NDN routing resources and the service chaining must be adapted consequently. Figure 4.5 illustrates such a process. To ease the understanding, we illustrate it in the context of a router named *SV-Router* which performs signature verification and whose load becomes consequently high. *SV-Router* periodically sends the number of entities in its *PIT* to the *VNFM* which in turn forwards it to the *NFVO*. Whenever a given threshold is crossed, the *NFVO* commands *VIM* and *NFVI* to scale-out the *VNF containers* to a specified number of replicas which is defined in the TOSCA profile. Once the deployment of the related virtualized network (VXLAN tunnel) and the *VDU* are finished, the *VNFM* retrieves the current configuration of the original *SV-Router*, which mainly consist of NDN routing information and sends the configuration to the replica of *SV-Router*. After the *NFVO* has

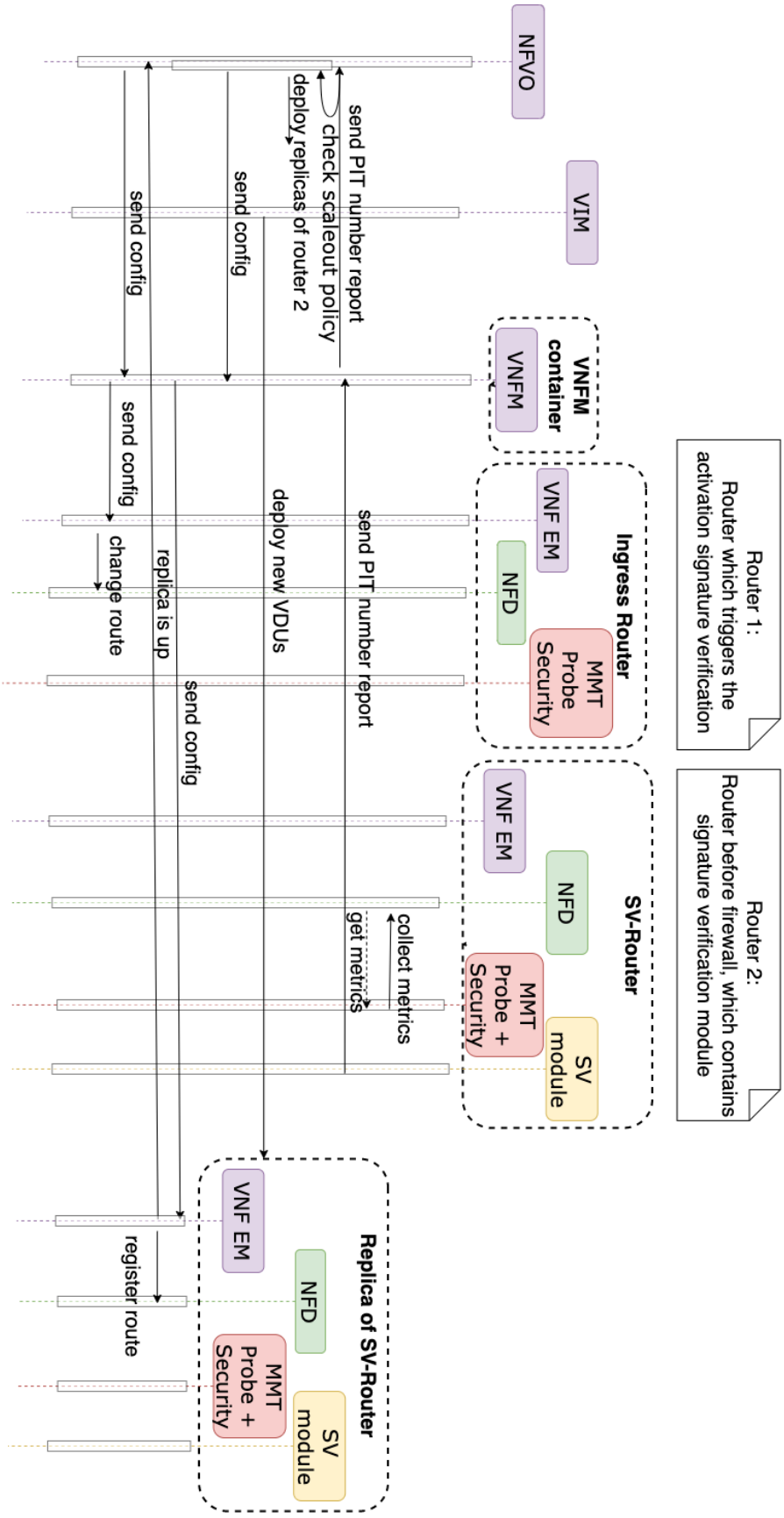


FIGURE 4.5: Sequence diagram of the scale-out policy firing

been informed that the configuration phase in a replica is finished, it instructs the downstream router of *SV-Router* to add a route to the replica. Finally, the forwarding strategy in this node is changed to *robin-round* which allows balancing the load into the different replicas of the *SV-Router*. As the scale-out process is performed progressively, it ensures that there is no downtime during the operation and the network continues to provide the service without any disruption.

4.6 Evaluation

In this section, we present the results we have collected in two different evaluation experiments of our content-oriented MANO framework. All experiments have been conducted in a virtual testbed hosted over Openstack in which each Point of Presence (PoP) is emulated by a virtual machine hosting an NFVI. And among these PoPs, one is selected to host our content-oriented orchestrator which dynamically spawns the VNFM as a Docker container. The first evaluation targets the deployment automation of an NDN virtual topology while the second evaluates to what extent our solution is able to dynamically reconfigure a virtual topology as well as virtual network functions according to events triggering policies. For each experiment, we first provide the selected topology and scenario, followed by a description and analysis of the results we collected. One can note that all presented results are the average of five repetitions bounded by 95% confidence intervals.

4.6.1 Deployment Automation

The evaluation of the deployment automation has consisted in measuring the deployment time of virtual components specified in a TOSCA specification according to different parameters which are the number of VNFs and their connection degree.

Use case

For all deployment automation evaluations, we have considered VNFs that host solely a containerized NDN router. Then, we have considered three basic topologies, depicted in Figure 4.6, each varying in terms of connectivity degree. The first one, *No connection*, only hosts VNFs without any virtual link. The second one, *Ring*, considers a standard ring network of n VNFs linked by $n-1$ connections, thus standing for a low connected network. The last one, *Mesh*, is a fully connected network exhibiting $n.(n-1)/2$ links, thus standing for a highly connected network.

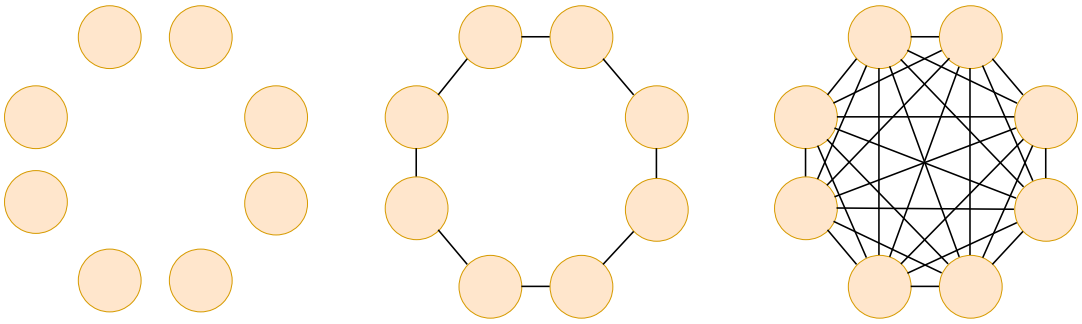


FIGURE 4.6: Selected topologies of NDN VNFs for the deployment automation (*No Connection*, *Ring* and *Mesh*)

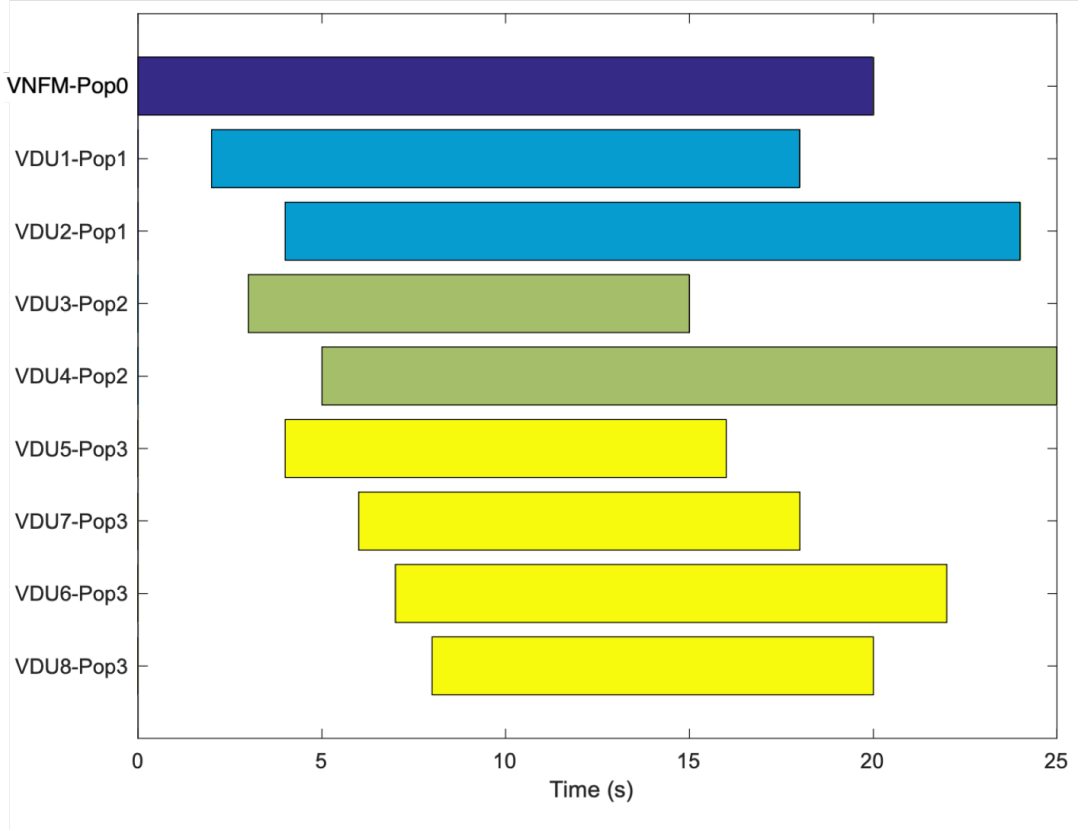


FIGURE 4.7: Snapshot of network deployment in case of 8 VNFs

Numerical results

Figure 4.7 contains the time series for the deployment process of 8 VNFs. Each color represents a PoP. The first PoP hosts the VNFM while three other PoPs host the 8 VNFs. At the infrastructure layer, the three PoPs start the deployment whenever they receive the related configuration from the NFVI. Afterward, the deployment of VNFs in each PoP starts concurrently, between each PoP and also within each PoP too. One can observe that the deployment time for each VNF is different since it only ends once the routing configuration of NDN components is achieved, which

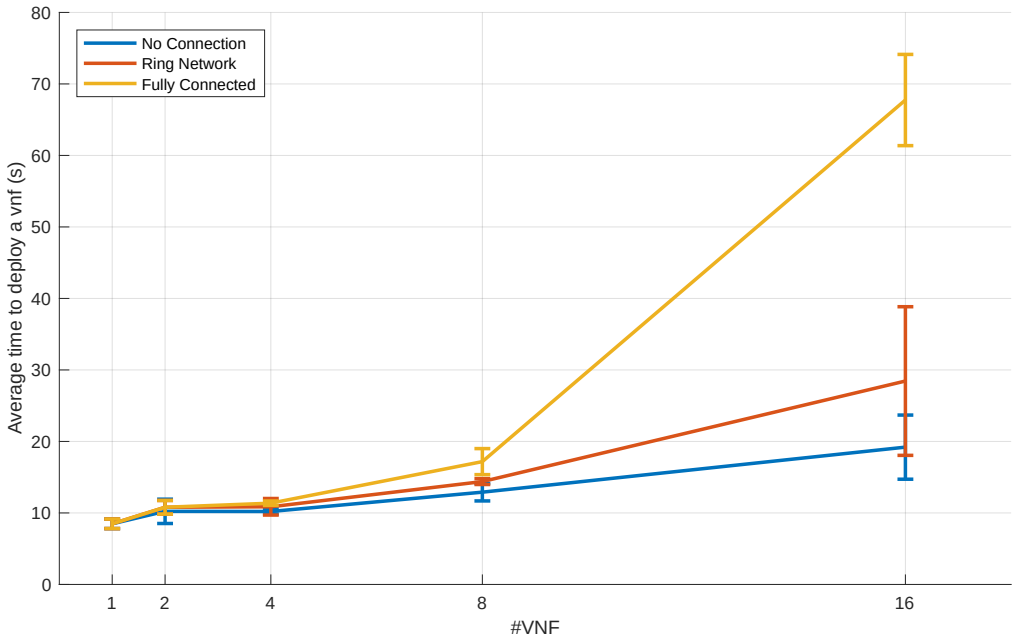


FIGURE 4.8: Average time to deploy a VNF

takes a variable time. In the following all measurements of deployment times follow a similar pattern.

Figure 4.8 depicts the average deployment time for a single VNF, according to the number of VNFs in case of our three selected topologies, while figure 4.9 depicts the overall deployment time. On these two figures, one can see that the deployment time is almost identical for a low number of VNFs (up to 8) and grows linearly. Over 8 VNFs, the deployment time differs according to the connection degree. In case of *No connection* and *Ring* topologies, the deployment time still follows a linear curve, while for the *Mesh* topology, the curve exhibits a exponential growth. This demonstrates the important cost of network configuration operations within the virtual network (virtual links and connections establishments), as compared to container spawning and underlying infrastructure configuration for the service chaining. For all deployment cases, the deployment time is acceptable with an average duration of 30 seconds for the *No connection* and *Ring* topologies and 90 seconds for the *Mesh* one.

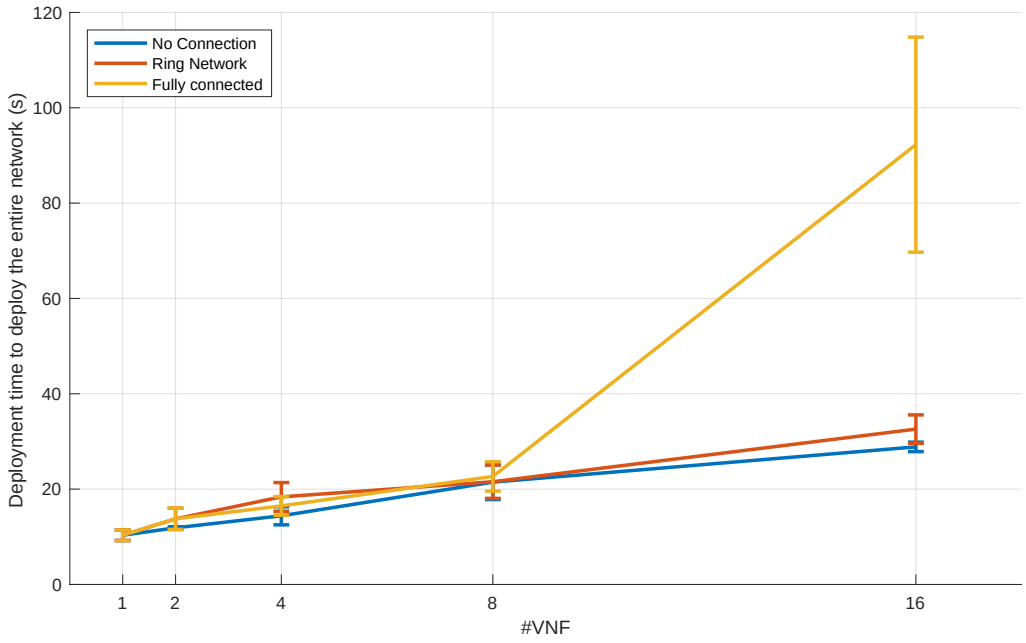


FIGURE 4.9: Time to deploy the entire network

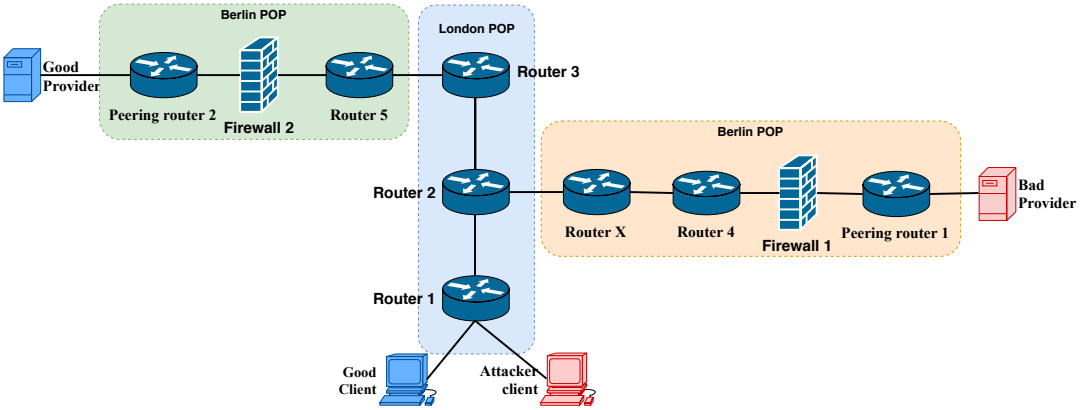


FIGURE 4.10: Experiment topology

4.6.2 Orchestration

As a second step of assessment, we evaluate the capability of our content orchestrated solution to dynamically react to events and its capacity to dynamically reconfigure the virtual network and its VNFs accordingly.

Use case

The scenario we consider addresses both performance and security anomalies which require the overall infrastructure to enforce dynamic configuration operations. The

topology we consider is depicted in Figure 4.10 which is an extract of the ClaraNet operator topology provided by the Internet Zoo Topology dataset [129]. Here, we consider two different VNFs, presented in Section III.B.1 which are NDN routers and NDN firewalls. Our three PoPs respectively host edge access nodes (in Berlin and Amsterdam), containing a peering node, a firewall and a router, and a core network (in London) containing three core NDN routers. A good producer of NDN content is connected to the Berlin PoP while a malicious one is connected to Amsterdam. The London PoP connects both a good user and an attacker whose purpose consists in corrupting the data hosted in all network caches to prevent the good user to access any desired content. This topology and scenario stands again for a CPA use-case, similarly to that of previous chapter.

To illustrate our extension of TOSCA for NDN, the two subsequent scripts respectively show how we specify NDN VNFs and an NDN forwarding graph.

The following script stands for the description of an NDN firewall VNF that blocks the NDN prefix */foo*, which is configured during VNF startup.

```
firewall_1:
     type: tosca.nodes.nfv.doctor.VNF.firewall
     properties:
       id: 6
       vendor: orange
       version: 1.0
       configuration:
         mode: accept
         rules:
           - action: append-drop
             prefix: [/foo]
     requirements:
       - VDU: VDU4
```

The next script depicts an extract of the content oriented forwarding information that provisions the routing in our use-case topology. This TOSCA node describes the full paths of the experiment topology shown in figure 4.10. It comprises three paths: the first one is from *Router 1* to *Router 2* via the connection points *VDU1_VL1_CP* and *VDU2_VL1_CP*; the second one is from *Router 2* to *Peering router 1* and the last one is from *Router 2* to *Peering router 2*. All of three paths allow the forwarding of *Interests* and receive *Data* for the */com/google* prefix.

```
   http_from_r1_to_r2:
     type: tosca.nodes.nfv.doctor.FP
     description: creates path for /http from r1 to r2
```

```
    properties:
      id: 1
      policy:
        type: NDN
        prefix: [/com/google]
      path:
        - forwarder: router_1
          capability: VDU1_VL1_CP

        - forwarder: router_2
          capability: VDU2_VL1_CP

  http_from_r2_to_as1:
    type: tosca.nodes.nfv.doctor.FP
    description: creates path for /http from r2 to as1
    properties:
      id: 2
      policy:
        type: NDN
        prefix: [/com/google]
      path:
        - forwarder: router_2
          capability: VDU2_VL10_CP

        - forwarder: router_x
          capability: VDUX_VL10_CP
        - forwarder: router_x
          capability: VDUX_VL2_CP

        - forwarder: router_4
          capability: VDU3_VL2_CP
        - forwarder: router_4
          capability: VDU3_VL3_CP
        - forwarder: firewall_1
          capability: VDU4_VL3_CP
        - forwarder: firewall_1
          capability: VDU4_VL4_CP
        - forwarder: peering_router_1
          capability: VDU5_VL4_CP
```

```
http_from_r2_to_as2:
  type: tosca.nodes.nfv.doctor.FP
  description: creates path for /http from r2 to as2
  properties:
    id: 3
    policy:
      type: NDN
      prefix: [/com/google]
    path:
      - forwarder: router_2
        capability: VDU2_VL5_CP
      - forwarder: router_3
        capability: VDU6_VL5_CP
      - forwarder: router_3
        capability: VDU6_VL6_CP
      - forwarder: router_5
        capability: VDU7_VL6_CP
      - forwarder: router_5
        capability: VDU7_VL7_CP
      - forwarder: firewall_2
        capability: VDU8_VL7_CP
      - forwarder: firewall_2
        capability: VDU8_VL8_CP
      - forwarder: peering_router_2
        capability: VDU9_VL8_CP
```

Orchestration scenarios

In order to evaluate the dynamic part of our orchestration framework, we consider three CPA mitigation scenarios: *No policy*, *No scale-out* and *Scale-out*. In the case of *No policy*, the entire network is deployed without any policy to mitigate security threats. This scenario is considered as a reference to evaluate the relevance of the subsequent ones. In the second scenario (*No scale-out*), the *NFVO* enables a policy that allows the dynamic configuration of the firewall and the dynamic activation of signature verification in NDN routers. Finally, in the *Scale-out* scenario, the *NFVO* enforces the mitigation policies presented above and it also enforces a scale-out operation on NDN routers which suffers from a too high computing load.

Table 4.1 summarizes the constant parameters that were used to run these three scenarios and the following scripts illustrate the policies that were used in the case of the *Scale-out* one.

The signature verification policy, described in the following script, is the one triggered by *Router 2* to activate the verification signature in targets *Router 4* and *Router 5* which are the best candidates to enforce a signature verification that can prevent corrupted content to be pushed by the bad producer.

```
- activate_signature_verification:
    type: tosca.policies.nfv.doctor.security.signature_verification
    targets: [router_4, router_5]
    triggers:
      peeringPoint1_verification:
        event_type: tosca.nfv.doctor.security.alert.cpa
        condition:
          constraint: triggered_by router_2
        action:
          action_type: update_router_mode
          mode: signing
          target_router: router_4
      peeringPoint2_verification:
        event_type: tosca.nfv.doctor.security.alert.cpa
        condition:
          constraint: triggered_by router_2
        action:
          action_type: update_router_mode
          mode: signing
          target_router: router_5
```

Although the above policy allows *Router 4* and *Router 5* to detect and report any corrupted content to the *NFVO*, the filtering operation is achieved by the firewall, and consequently a dedicated firewall policy is specified in our NDN TOSCA profile. Specifically, the policy illustrated in the next script is the one triggered by *Router 4* and *Router 5* and it allows the configuration of *Firewall 1* and *Firewall 2* to be dynamically augmented with a rule to block the corrupted content.

```
- update_firewall_1:
    type: tosca.policies.nfv.doctor.ndn.security.update
    _firewall
    targets: [firewall_1, firewall_2]
    triggers:
```

```
    peering_point_1:
      event_type: tosca.nfv.doctor.security.alert.poisoned_content
      condition:
        constraint: triggered_by router_4
      action:
        action_type: update_firewall
        target_firewall: firewall_1
    peering_point_2:
      event_type: tosca.nfv.doctor.security.alert.poisoned_content
      condition:
        constraint: triggered_by router_5
      action:
        action_type: update_firewall
        target_firewall: firewall_2
```

Finally, the following script provides an example of performance policy which allows scaling out the VNFs *Router 4 Router 5* to three replicas whenever a given threshold is crossed. In the NDN case, this is the number of entries in the PIT of NDN routers.

```
scaling_out_policy:
  type: tosca.policies.nfv.doctor.ndn.scaling
  targets: [router_4, router_5]
  triggers:
      scale_out:
      meter_name: PIT
      event_type: tosca.policies.nfv.doctor.ndn.n_pit
      condition:
          constraint: pending_interests greater_than 5
          threshold: 5
          comparison_operator: gt
          period: 10
      action:
          action_type: scale_out
          number: 3
```

Figure 7 exhibits time series of *Good Data* (blue lines), *Bad Data* (red lines) and *Lost Data* (black lines), expressed in terms of number of packets per second, received by the good user in the context of the three scenarios described above.

In the *No policy* scenario, the frequency for *Good Data* is approximately 10 packets/s. However, there is also a substantial amount of *Bad Data*, about 3 packets/s.

TABLE 4.1: Experimental constants

Constant	Value
# Good producer contents	10000 contents
# Bad producer content	1000 contents
Data's freshness period	4 sec
Good producer link latency	100ms
Bad producer link latency	10ms
User's default *Interest* rate	10 interest/sec
Attack's *Interest* rate	32 interest/sec
Sampling period	5 sec
Experiment duration	10 minutes
Number of replicas in Scale-out scenario	3
Number of entries in the PIT to trigger Scale-out	5

Besides, some packets are lost in the communication, but this amount is negligible. This scenario demonstrates that without mitigation, the attack can corrupt up to one third of the content provided to a good user.

Numerical results

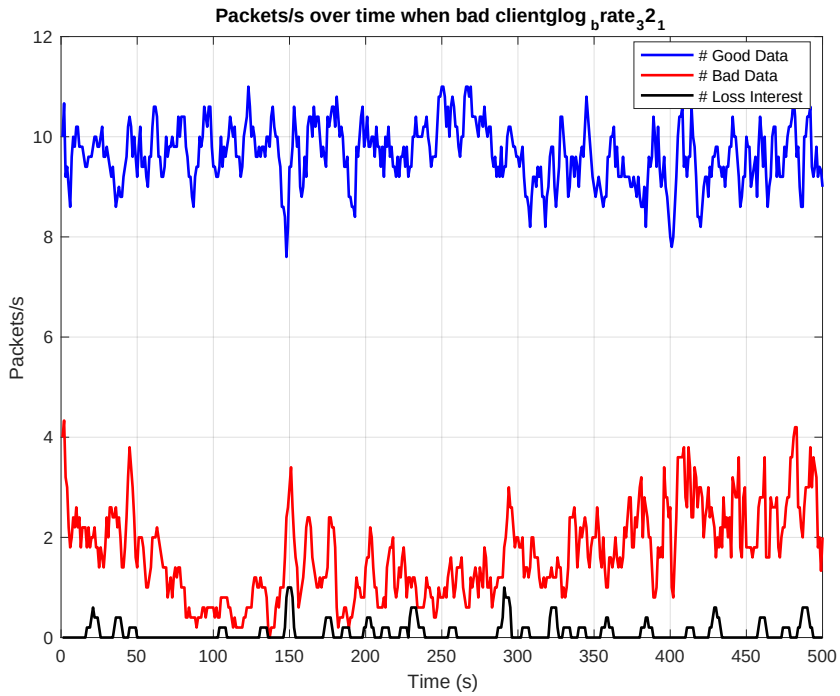


FIGURE 4.11: Snapshot of Good/Bad/Lost data in the No Policy scenario

In the *No scale-out* scenario, the green line shows the time at which the network detects the attack and triggers the firewall and signature verification policies to mitigate the attack. We can observe that after this moment, the number of *Bad Data* abruptly drops to zero. However, the number of *Good Data* also decreases slightly, and the number of *Lost Data* increases remarkably. The reason of this phenomenon lies in the overload the routers performing the signature verification suffers from, which prevents it from reliably achieving its basic forwarding operations.

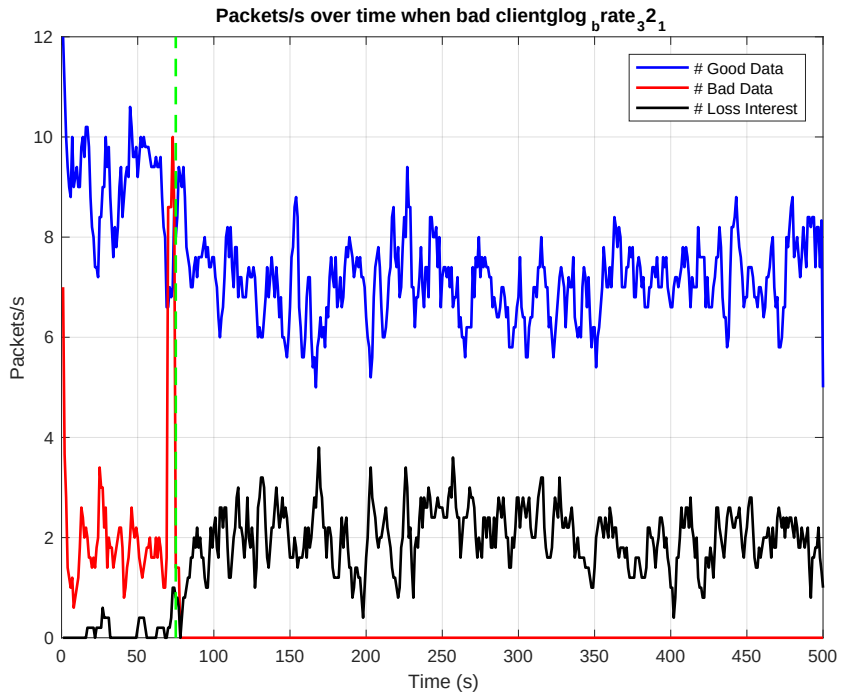


FIGURE 4.12: Snapshot of Good/Bad/Lost data in the No Scale-out scenarios

In the last scenario, *Scale-out*, the two dark lines of figure 4.13 show the instant at which the scale-out configuration operations start and terminate. In this scenario, one notices that the mitigation efficiency is identical to the *No scale-out* scenario, with an excellent filtering of corrupted data. Moreover, when the scale-out is enforced, the network returns to a normal state: *Good Data* gets back to 10 packets/s, and *Lost Data* drops to a negligible value. We can also observe that during the period of scale-out, there is no downtime of the network.

Beyond these illustrations of mitigation, over all our experiments, the percentage of *Bad Data* in the *No Scale-out* and *Scale-out* scenarios are 5,1% and 4,8% respectively, which are smaller than in *No Policy* (14,1%). This result demonstrates the efficiency of both the signature verification and the firewall reconfiguration policies.

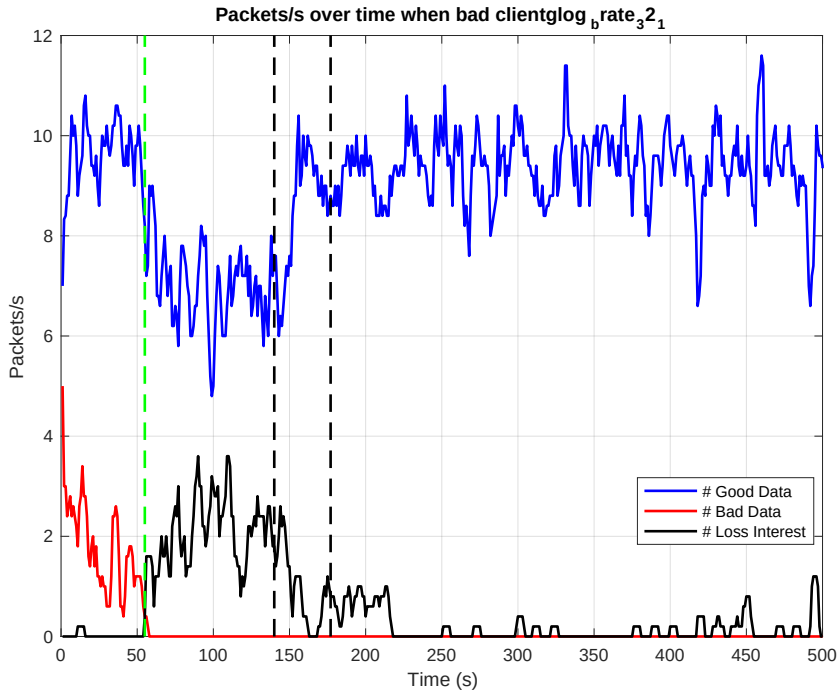


FIGURE 4.13: Snapshot of Good/Bad/Lost data in the Scale-out scenario

The percentage of *Bad Data* which remains on these two scenarios is the percentage of *Bad Data* before the attack detection. After the activation of these policies, the number of *Bad Data* drops to zero. Figure 4.14 emphasizes this statement by illustrating the ratio of Bad / Good data with different attack rate. As has been noted, the percentage of *No Scale-out* and *Scale-out* are significantly smaller than the percentage of *No Policy* especially when the attack rate is high.

On the other hand, the percentage of *Lost Data* in the *Scale-out* scenario is 5,5% which is smaller than in the *No Scale-out* one (15,2%). More precisely, in the *Scale-out* scenario, the percentage of *Bad Data* and *Lost Data* before the activation of the scale-out policy are 13% and 6,1% respectively. However, after the scale-out operation, these two metrics decrease to 0% and 3,8% respectively, which proves the relevance of the scale-out policy. Figure 4.15 depicts in detail the total of loss packets in three scenarios. As can be seen, number of *Loss* of *No Policy* is the smallest among the three scenarios as the SV module is not activated. In case of low attack rate (smaller than 8 Interests/s), the number of *Loss* of *Scale-out* and *No Scale-out* is also ignorable as the *NDN router* can still handle it. Nevertheless, when the attack rate increases, it exceeds the capacity of the *NDN router*, hence has a significant impact on the packet processing in *SV-Router* and the number of *Loss* of *No Scale-out*. Also, in *Scale-out*, the

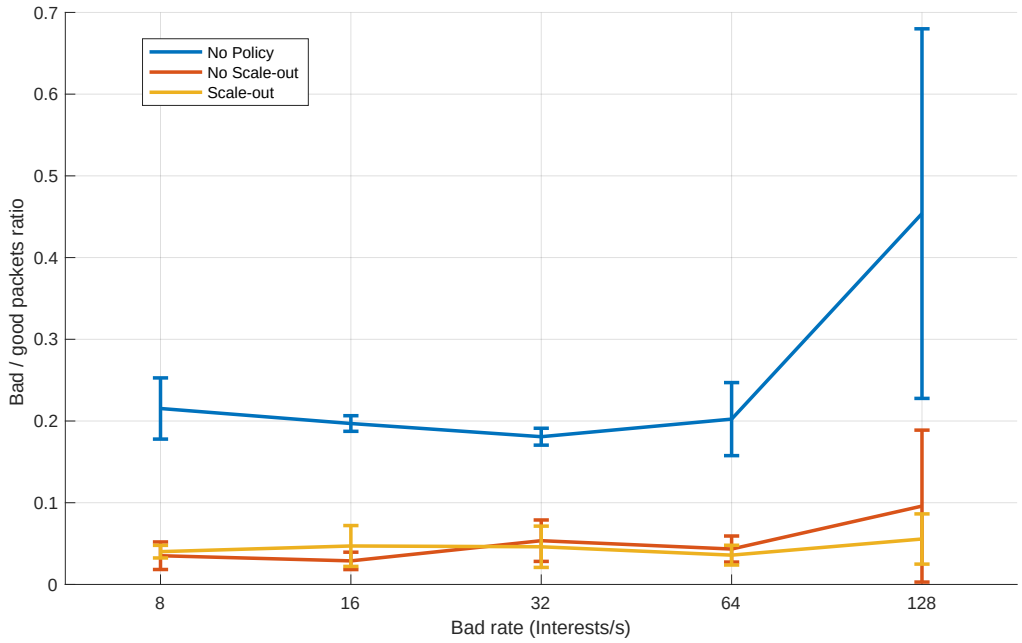


FIGURE 4.14: Bad/Good Data ratio

number of *Loss* is highest when the attack rate is 32 Interests/s. The reason is that the threshold to trigger the scale-out process is the same for each attack rate. With a high attack rate, the scale-out can be activated ahead, hence, reduces the number of *Loss* immediately. The efficient of *Scale-out* is shown also in Figure 4.16, which illustrates number of *PIT* of *SV-Router* before and after the scale-out. After the scale-out, the number of *PIT* entities is smaller than before. With a low attack rate (8 Interest/s), there are almost no differences between the number of *PIT* entities before and after the scale-out, however, the difference between two stages enlarges when the attack rate increases. The reason for this phenomenon logic is similar to the case of the number of *Loss*, the trigger of the *scale-out* policy is easier to activate with a higher attack rate, hence performing the scale-out process sooner than the scenario of a small attack rate.

Finally, figure 4.17 illustrates that the delay in detecting and shut down the attack in *Scale-out*. Generally, as the forwarding strategy in *Router*2 is multicast, the delay in detecting the attack is nearly the same [127]. Also, the delay in shutting down attack decreases slightly when the attack rate increases, since with a high attack rate, the chance of bad data in top rank content increases, thus, the poisoned content is become more comfortable to block.

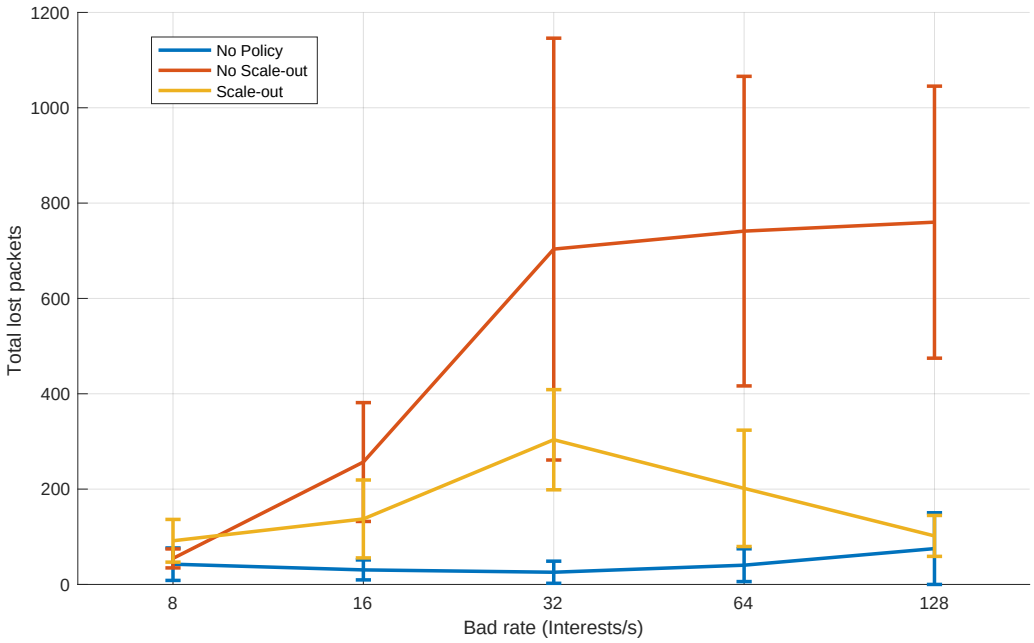


FIGURE 4.15: The total number of loss packet

In the final analysis, the numerical results demonstrate that both policies to remediate the attack and the performance incidents are efficient. There is still room for improvement, especially regarding the condition to activate the scale-out policy and the delay to trigger the remediation process when the attack rate is low.

4.7 Conclusion

Deploying NDN is a strategic and complex task which can be supported by a combination of SDN/NFV and smart orchestration. Besides, remediation deployment is an obligation for the security content-oriented orchestrator. In this chapter, we have defined the core elements for the design and implementation of a content-based orchestration for virtualized NDN networks. We have shown that the TOSCA standard can be extended to become a solid content-based service specification template. With novel orchestration components, we have demonstrated that a content-oriented orchestrator can both automatically deploy a virtual NDN topology, but also enforce dynamic reconfiguration in the virtual network triggered by security and scale-out policies. We have assessed the architecture of real deployment and collected both functional proof and measurements of the viability of the proposed system.

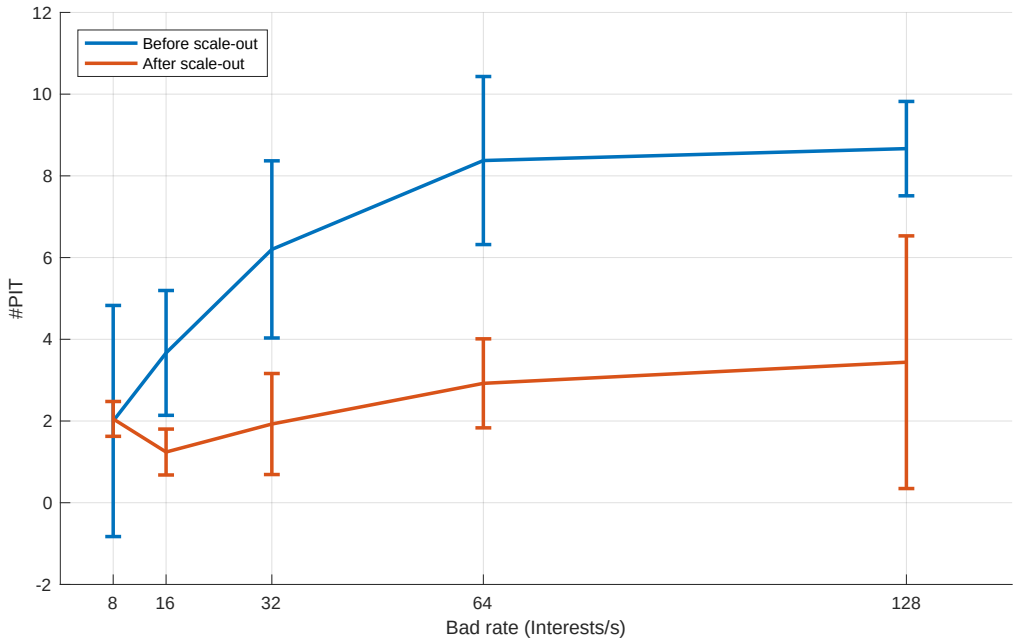


FIGURE 4.16: Number of entities in PIT before and after scale-out

One drawback that can be seen in the proposed security content-oriented orchestration is that the performance of the detection module depends on both the computation capacity of NDN router and the attack rate. We deliberately deactivate the signature verification module in NDN nodes in order to save the energy and increase the performance of these nodes. There is no reason that the anomaly detection module runs all the time since it consumes a significant amount of computation resources. Also, if an attacker performs the attack on multiple nodes at the same time, triggering the scale-out on multiple nodes can lead to a run-out of computation capacity in the infrastructure layer. If we deactivate this module by default, the number of loss packets will be noticeable and the delay in detecting the attack would increase notably and could give the attacker enough time to finish their purpose. Indeed, this drawback is shown in the Evaluation section, which describes that with a constant capacity of computation and without scale-out policy, the signature verification module in an NDN router cannot handle a large number of Interests and the total number of loss packet is enormous. To address the challenge, in the next chapter, we will propose a content-oriented anomaly detection architecture that can support distributed anomaly detection efficiently by leveraging the computation sharing between nodes.

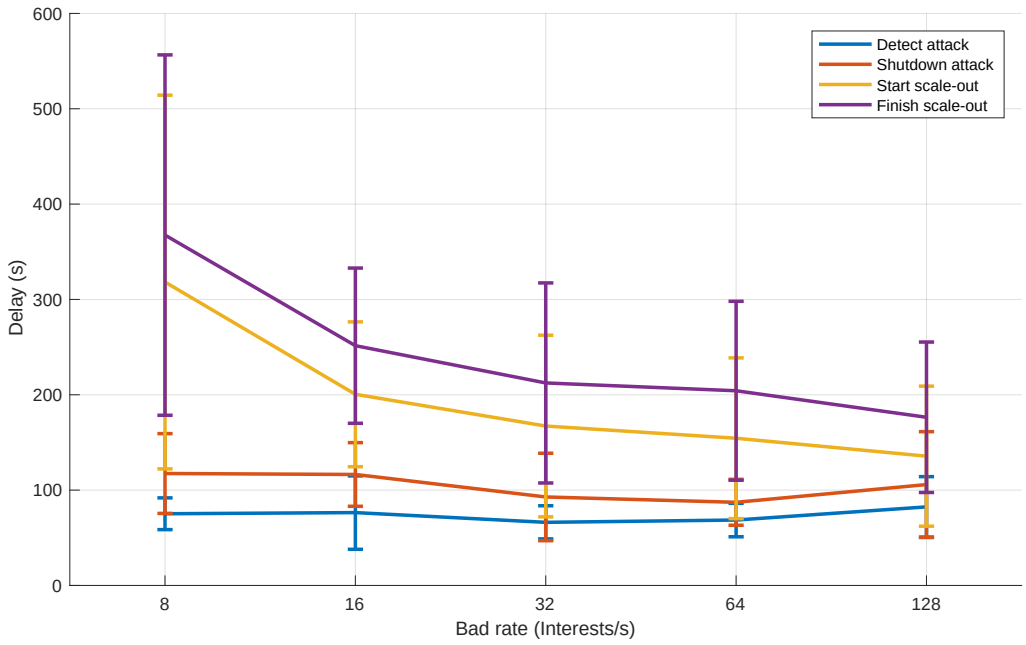


FIGURE 4.17: Delay to detect, shutdown attack and start process scale-out

Chapter 5

A Content-Oriented Anomaly Detection Architecture

Contents

In this chapter, we will address the last challenge concerning the security plane, which is the performance of security functions and how to optimise the use of computing resources while performing them. A security plane must not only detect the

attack but also perform these detection functions effectively by controlling the use of computing resources made critical due to the complexity of the security functions and the resource constraints of the nodes. To tackle these issues, the content-centric paradigm, enhanced with computing features, offers a promising solution to reduce the computation resources and improving the scalability of the anomaly detection functions. As a result, we come up with the proposal of a content-oriented security plane for NDN. We first indicate and verify several hypotheses, which are the root of our concept. Secondly, we present the overall architecture of the proposed approach. Thirdly, we present our main contributions, which are the adopted naming scheme and data structure, and the transformation of functions into λ-calculus. In order to evaluate the proposed algorithm, the use case of the Content Poisoning Attack still is considered, and the numerical results shown in the last section demonstrate the effectiveness of our proposed algorithm. The contribution in this chapter has been presented in [130].

5.1 Background on Named Function Networking

Inspired by the concept of both active networking and the content-centric communication model [131], the Named Function Networking (NFN) architecture, introduced in [132,133], further extends this concept to resolve names to computation functions. In NFN, the functional programs are expressed in λ-calculus [134], encoded in hierarchical CCN names that can be aggregated and carried in *Interest* packets.

5.1.1 How NFN works

The operation of NFN relies on a λ-calculus expression resolution engine in each NFN node, and optionally an application processing or the compute server can also be hosted in an NFN node. The routers use the longest-match name lookup to forward *Interests*. If an NFN node hosts a compute server, its named function is published as a λ-calculus expression in the network, and *Interests* can be forwarded to the function host. If a copy of a function result is cached at an intermediate node, it will be returned to the requesting user along the reverse path. Otherwise, one of the NFN nodes, which has received the *Interest* on the path to the data source, will attempt to compute the result based on the policies and the available processing resources.

For example, a user sends an *Interest* to request the number of words of a content, *i[λy /name/of/wordcount y) /name/of/content]*. *Interest* is transferred to the source of *Data/name/of/content* using the longest prefix rule. If a copy of the result exists cached on a node in the source of *Data/name/of/content*, then the result is returned to

the requesting user on the reverse path. Otherwise, one of the NFNs that received the *Interest* on the path can calculate the result based on the policies and the availability of its processing resources. The node can extract the names of the expression λ in the *Interest* and separate it into individual *Interests*, for example, for */name/of/content* and */name/of/wordcount*. If the content *Interest* and code for the *wordcount* function is retrieved, the node performs the calculation, returns the result, and caches it to serve other requests. If no entity matches the initial *Interest* to provide the result, the NFN node forwards the *Interest* to the code source to perform the calculation. If the computation is served on the path to its source, the result travels the reverse route to the proxy point, and then returns to the original requesting user. In general, an NFN node can evaluate and divide a named expression, and retrieve code, data, intermediate results, or distribute tasks by sending *Interests* with subexpressions to different nodes. It thus completes the calculation and sends the response to the original requesting user.

Some works have proposed extensions to NFN, in particular to better control calculations or provide them as a service. In its basic mechanisms, NFN is only able to start a calculation and wait for a result. In [135], the authors introduced a mechanism to drive long duration computations on an NFN network using the concept of R2C (Request-to-Computation) which allows users to control the computation when it is executed at the same time inside the network. In addition, the authors of [136] propose an extension of the architecture of NDN with NFN as a service, where the network is considered as a computer and the computation approaches the edges of the network. They introduce new components, such as the Kernel Store, which stores the code of functions and also decides which functions to perform in order to migrate them closer to the user in an ICN environment without a global view of the network. The source code is available for the research community (https://gitlab.com/mharnen/NFaaS).

Several implementations of NFN coupled with reference implementations of ICNs are available. CCN-lite (https://github.com/cn-uofbasel/ccn-lite) is a small and lightweight implementation of the CCNx and NFN protocols. It is written in C and supports multiple platforms. A Python version also exists called PyCN-lite (https://github.com/cn-uofbasel/PyCN-lite), dedicated to IoT environments, and which supports both NDN and CCN protocols. Dedicated implementations for NFN also exist, for example, we cite NFN-Scala (https://github.com/cn-uofbasel/nfn-scala) which is written in Scala, interfaceable with CCNLite and allows to write λ-calculus programs. Finally, PiCN (https://github.com/cn-uofbasel/PiCN) is written in Python 3 with a modular ICN implementation designed to deploy NFN applications.

5.1.2 NFN applications

A typical use of NFN consists in performing data processing in the network. In [137, 138], the authors use NFN for the orchestration of treatments on multimedia content and for the processing of data streams in the network. NFN has also been proposed as a query language [139] to improve the naming scheme for content in Information-Centric Networking (ICN). An In-Network Access Control for NDN is proposed in [140, 141] which also utilizes NFN as Data Processing Unit (DPU) between clients and providers.

NFN has also found its application in an Edge/Fog computing context. The function program can be stored in cache of nodes and migrated to the network according to the request of the users [136]. A study on NFN as an architecture for Mobile Edge Computing (MEC) systems is proposed in [142]. It shows how to identify network resources and to support the execution of their functions [143]. The authors in [144] have also explored the possibilities of an augmented reality network using NDN with NFN as an associated processing mechanism. In [145], authors propose another ICN-based service platform at the edge of the network with NFN as a protocol for managing service interactions.

In the context of the Internet of Things, several solutions have been proposed to exploit the mechanisms of NFN. The authors of [146] propose a management of computing services to assign and schedule computational tasks in an IoT network. In [147], the authors also use NFN in an IoT network where sensor data and processing functions are stored. In [148], the authors also propose an architecture and a naming scheme to cache data and processing within an IoT network. The authors of [149] also use NFN as the DPU for the IoT coupled to NDN. In [150], Lenord et al. use NFN in an IoT gateway to propose a new routing strategy in NDN.

5.1.3 NFN for security

Few works have proposed NFN as a mechanism associated with security applications. We identified the work in [140], where the authors propose a network access control mechanism for NDN using NFN as a computation mechanism between the client and the provider for the exchange and verification of the symmetric key between them. Each NFN-enabled node is equipped with RSA public-private key pairs to perform access control functions on data, while the NDN node functions as a data storage unit to serve ACLs and corresponding keys. Another work [151] proposes more generally a supervision protocol, for data centers, based on ICN, with an NFN-compatible processing mechanism. This protocol includes in a *Interest* a parameter that designates the identifier of the data replica to be monitored. It operates with

several phases to identify these replicas in the network to then include in the requests specific functions to collect the monitoring data (CPU, load, memory) of their associated servers.

In brief, NFN has drawn much attention from the research community in recent years, with different use-cases, in the context of NDN, IoT, and Fog computing and in the following, we explore the opportunity to consider it in a security context, for anomaly detection. We especially consider Bayesian Network inference which is commonly used as a substrate.

5.2 A content-oriented anomaly detection architecture

Since a security plane continuously monitors a large-scale network, a large number of security operations are repeated. In particular, it appears that each node performs the same security tasks over time, especially in a normal situation where there are no attacks or significant changes. Furthermore, some of the outputs of security functions most probably have already been calculated by neighboring nodes.

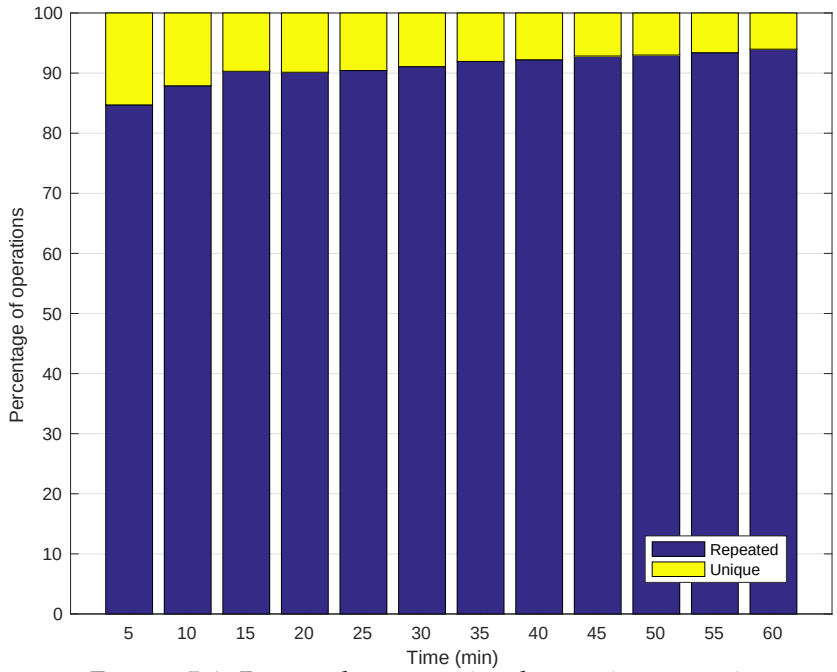


FIGURE 5.1: Repeated computational operations over time

We verified this hypothesis by measuring the percentage of repeated operations. For this, we used the hash of the name of the function and its parameters; for example, hash(f,x) where f is a function with parameter x. Since the name of a function with its parameters (i.e. inputs) identifies the output, the percentage of repeated values of hash(f,x) reveals the percentage of repeated operations. As shown in figure 5.1, we generate normal traffic in a real NDN testbed during one hour with three

NDN routers. We found that in a normal NDN operation, 85%-95% of computational security operations repeated the same functions with the same parameters. We also observe that at the beginning the percentage of repeated operations is lower than later. The reason for this is that the metrics follow a normal distribution, and are therefore more likely repeated gradually over time. This property verifies our hypothesis and demonstrates that there is still room for the improvement of the security functions by reducing the repeated executions of the same computations over time.

With this in mind, we leverage the in-network caching feature of Named Function Networking, which offers a substantial way to reduce the computation resources of security functions. As a consequence, in the proposed architecture, the anomaly detectors are decomposed into anomaly detection functions, as illustrated in Figure 5.2. These functions demand significant resources to be executed. And since a λ-calculus [134] expression resolution engine is integrated into each NFN node, to adapt the inference algorithm in NFN, we transform the functions of the VE algorithm into λ-calculus, hosted in the NFN processing server. Afterward, these functions are connected with an NFN forwarder, which forwards the computational request or caches the result. The rest of architecture is the same as the architecture presented in Chapter 3.

As explained in chapter 2, we consider Bayesian Network (BN) inference as an anomaly detection framework since it stands for a representative function that numerous security components consider. However, NFN is still in the development stage and processes only a few types of data with a limit in the size of packets. Hence, to transform the inference algorithm so that it can be distributed in the NFN network, first and foremost, we must propose a naming scheme and data structure for the objects used by the inference algorithm, as well as transform the inference algorithm into λ-calculus functions. These two challenges will be addressed in the following sections.

5.3 Naming scheme and data structure

As depicted in the VE algorithm, presented in section 2.3.1, the $factor$ is the core of the inference algorithm. A $factor$ is not only the input but also the output of the functions. Besides, the *evidence* parameter can also be used to calculate the *factor reduction*. A variable is also a parameter in the VE algorithm but it does not carry additional information except its name. As such, we deliberately do not name the variable and use it as a string. The structures for $factor$ and *evidence*, their naming schemes and the naming scheme of functions are described in the following section.

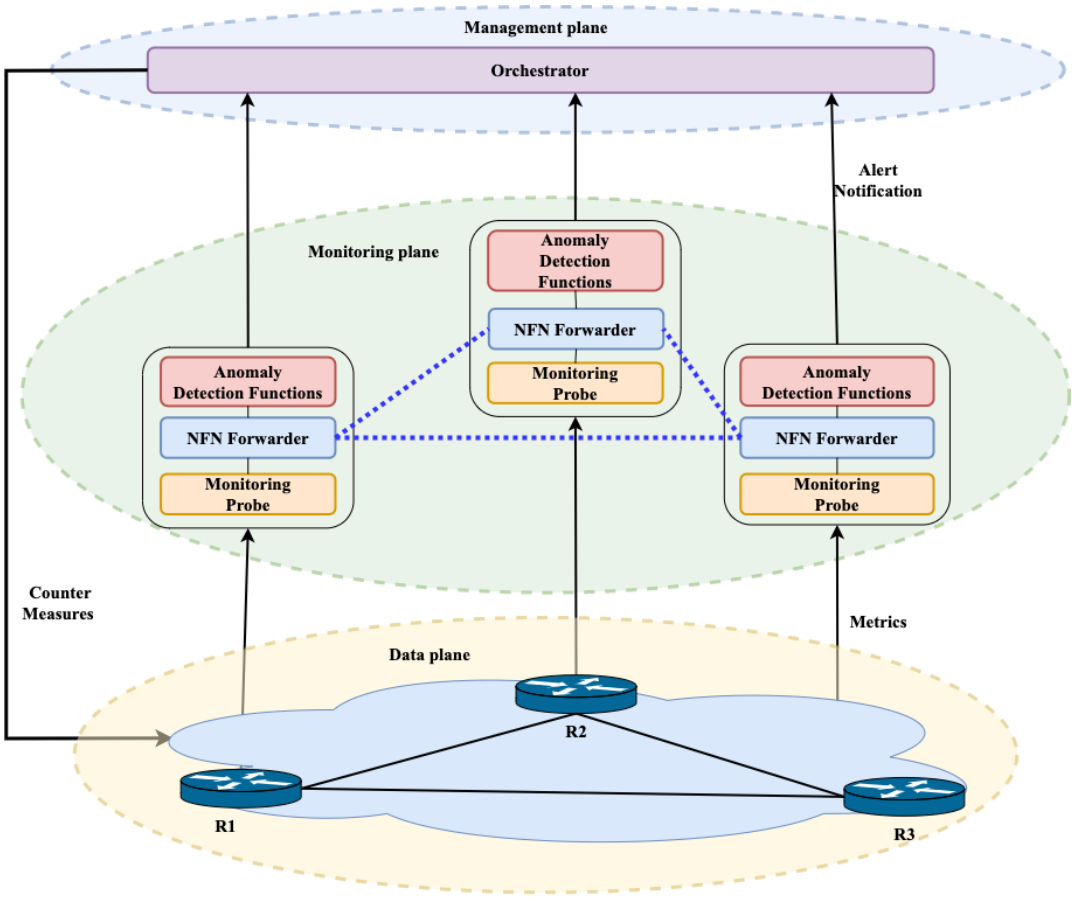


FIGURE 5.2: The overall architecture of the proposed content-oriented anomaly detection.

5.3.1 Factor

As has been pointed out, a *factor* encompasses the list of random variables, their dimensions, and the values of ϕ. We define the data object and data structure for our algorithm, as a factor could be named as one or multiple data objects depending on the naming scheme. We list two possible options for the naming scheme and data structure of factor and motivate our choice.

In the first option, each factor is formed of multiple data objects and each data object is an entry. The resulting data structure will include the names of random variables, their values correspond to the entry, and the value ϕ of the entry. The advantage of this first option is the simplicity of transforming the functions into lambda calculus, as each data contains only one value that corresponds to the value of the entry. However, there are several drawbacks. Firstly, a factor has multiple entries and the entry is not repeated between two different factors. Hence, a noticeable amount of data must be stored. Secondly, the value of the entry, which is only a number, is the unique part that will be used in the numerical operations. While

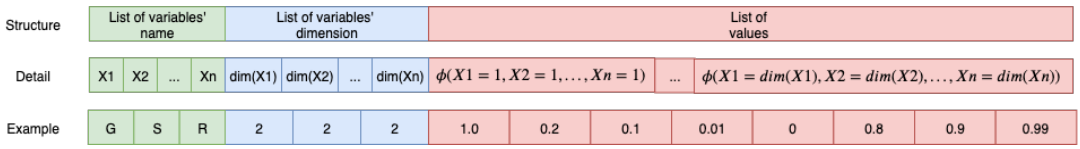


FIGURE 5.3: Data structure of a $factor$

the first two parts, which are the names and the values of the random variables demand an abundant space in the memory. Finally, since a factor operation uses all the factor's entries, a large number of parameters will be required to feed the operation, thus making the approach more complicated.

In the second option, the $factor$ packet includes three parts: the list of the names of random variables, their dimensions, and the list of all values of ϕ. The two first parts refer to the meta-data of the packet, while the third part is the actual data. The list of all values is sorted in the order of a $factor$. In other words, the third part of the data structure is a serialization of the values of ϕ where the $factor$ is expressed in the form of a table. Figure 5.3 illustrates this structure. In this option, the number of data objects stored is significantly smaller than in the first option as it stores a factor as an object, not an entry as an object. Moreover, the proportion between the meta-data and the actual data of the packet is equivalent, and the more complex factor is, the more we reduce this proportion. Hence, this allows reducing the necessary storage and bandwidth. It also reduces the number of parameters needed by the operation. The second solution is not the sole alternative, but it is the simplest and the most suitable one to name a factor.

More specifically, our proposed data structures include two types of $factors$: *initial factors* and *temporary factors*.

The *initial factors*, which are established by CPDs, are named as follows: */data/fac/initial/<name of variables>*. As an example, */data/fac/initial/GzSzR*, with "z" being the separator character, is the name for the $factor$ $\phi_3(G, S, R)$ shown in Table .

$factors$ are also the outputs of operations in the inference algorithm and then the inputs for the subsequent operations. Therefore, *temporary factors* are used to distribute the computation between nodes in an NFN network. These $factors$ are named using their hash: */data/fac/temporary/<hash(factor)>*. An alternative for naming a *temporary factor* consists in using its values as its name. However, when it is complex, the name may become too long to be stored in an NFN packet. Another choice consists in generating the name of the *temporary factor* randomly or incrementally. However, if we name the $factors$ in each NFN node randomly or incrementally, when a node asks others to compute an operation and another node already uses this name for a different $factor$, the result will be inexact. Hash functions avoid such a collision since the same hash means that the same data is stored in the cache.

5.3.2 Evidence

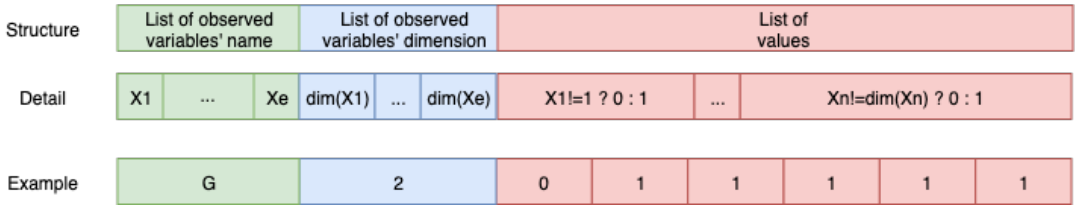


FIGURE 5.4: Data structure of the *evidence* $G = 2$ in the context of factor $\phi_3(G, S, R)$

Similarly, a packet of *evidence* encompasses three parts: the name of variables, their dimensions, and the values of the *evidence*. The data structure of *evidence* is designed considering the following ideas. Firstly, the *evidence* of a variable $X_e = x_e$ shows that the value of the variable X_e is known (x_e). As a consequence, entries that encompass values $X_e \neq x_e$ will be dropped while introducing the *evidence*. For this purpose, the value of the *evidence* for a variable in a packet consists of a chain of 0's and 1's. The value 0 signifies that the corresponding entry will be dropped while value 1 means that the entry will remain while introducing the *evidence*. As an illustration of the *evidence* $G = 2$, the value in the *evidence* packet is 01.

Secondly, an *evidence* doesn't consist of only one variable. Rather, it is composed of a multitude of variables. Consequently, the data of the *evidence* is the merging of the values of each variable in the *evidence*. As an illustration of the *evidence* $G = 2, S = 1$, the value in the *evidence* packet is 01z10.

Finally, an *evidence* is designed to be computed with a $factor$. Some variables belong to *evidence*, and other variables belong to $factor$. They both should be present in the value of the *evidence*. The values of these variables are still unknown. This means that they will remain while introducing the *evidence*. Hence, they are marked by a chain of values 1. For example, the value of the *evidence* $G = 2$ in the context of the $factor$ $\phi_3(G, S, R)$ is 01z11z11. Figure 5.4 illustrates the data structure of the *evidence* $G = 2$ in the context of factor $\phi_3(G, S, R)$.

On the other hand, the meta-data encompasses only names and dimensions of variables that belong to the *evidence* E. As an example, for the *evidence* $G = 2, S = 1$ in context of the $factor$ $\phi_3(G, S, R)$, the meta-data of this *evidence* is $GzSw2z2$, while the full packet of the *evidence* is $GzSw2z2w01z11z10$, with "w" and "z" being the separator characters.

In the proposed approach, an *evidence* is named using the list of names of variables and the value of the *evidence*: */data/evi/<name of variables>/<values of evidence>*. As the *evidence* should be customized to be operated with $factor$, we also deliberately list the variables in the $factor$ that are still unknown and marked as "NaN". For

example, */data/evi/GzSzR/2zNaNz1* is the name for the *evidence* $G = 2, S = 1$, which is compatible with $factor\ \phi_3(G, S, R)$.

5.3.3 Function

As mentioned in chapter 3, the proposed anomaly detector is a probabilistic-based method, which requires a noticeable computational capacity to execute the anomaly detection functions. We argue that by distributing and sharing the anomaly detection functions between nodes, it is possible to reduce the computational time as well as the resource consumption.

To this aim, the final step of the naming scheme is to name the anomaly detection functions. Firstly, prefixes */func/<name of functions>/* are reserved for NFN functions. To conduct the inference, various functions are needed but only the four main ones of the VE algorithm are named in the NFN forwarder. Other functions, called helper functions listed in table 5.1, are also transformed into λ-calculus and added in the NFN compute server, but they are not named using the previously defined naming scheme. These functions are identified by their names and parameters (where a parameter can be a string, integer, or a data name). For instance, the functions of the VE algorithms are:

- */func/reduce/(/data/fac/...,/data/evi/...)*
- */func/product/(/data/fac/...,/data/fac/...)*
- */func/marginalize/(/data/fac/...,variable)*
- */func/normalize/(/data/fac/...)*

To illustrate our naming scheme, we describe an example of calculating the probability in the example "rain, sprinkler, and grass". We propose a simplistic topology consisting of two NFN nodes and one NFN client: NFN Node 1 and NFN Node 2. The NFN Node 1 hosts two functions, /func/reduce and /func/normalize/, while the NFN Node 2 hosts two functions, /func/product and /func/marginalize/. The NFN client node sends NFN requests to NFN Node 1 and NFN Node 2. Figure 5.5 shows the sequence diagram of the calculation of $P(R|G = 1)$ and $P(R|G = 2)$ in the context of the example "rain, sprinkler and grass".

In the first request, we send a computation request $P(R|G = 1)$ to the NFN client. Then, based on the VE algorithm, the NFN client sends a computation requests to NFN Node 1 and NFN Node 2. The first step in the VE algorithm is *factor reduction*. In the example, we have one sole evidence $G = 1$, the initial factor in this request is /data/fac/initial/GzSzR while the evidence is /data/evi/GzSzR/1zNaNzNaN.

The second step consists of calculating the *factor product* $\phi_1(R)\phi_2(S,R)\phi_3'(S,R)$. To calculate the *factor product* of several $factors$, we perform the factor product of one $factor$ to the next one and then perform the same operation on the result, which is also a $factor$, and do this successively until we have only one $factor$. More specifically, the request /func/product(/data/fac/initial/R,/data/fac/initial/SzR) corresponds to the *factor product* of $\phi_1(R)\phi_2(S,R)$. Next, the result of this request together with the results of factor reduction are the parameters for the *factor product* $\phi''(R,S) = \phi_1(R)\phi_2(S,R)\phi_3'(S,R)$, which is named /data/fac/temporary/e4a15c5f. The third step consists of eliminating the variable S in the $factor$ $\phi''(R,S)$ to get the factor $\phi'''(R)$, which is named /data/fac/temporary/e4a15c5f. Finally we perform the *factor normalization* for the $factor$ /data/fac/temporary/e4a15c5f to get the conditional probability $P(R|G=1)$.

In the second request, we change the *evidence* from $G = 1$ to $G = 2$. Similar to the first request, we also perform the first step, the factor reduction. However, the parameter is now different. Therefore, we perform the factor reduction with the same initial factor, but with the evidence being different, which is /data/evi/GzSzR/2zNaNzNaN instead of /data/evi/GzSzR/1zNaNzNaN. In the second step: the *factor product* is /func/product(/data/fac/initial/R,/data/fac/initial/SzR). Thanks to the cache we do not need to re-execute the computation even if the computational request is different. The next steps of this second request are the same as in the first request.

As can be seen, the aforementioned example demonstrates that the computation can be distributed in the network and can leverage the cache to reduce the computational time.

5.4 Transformation of functions into λ-calculus

In the following section, we will explain in detail how we transform these functions into λ-calculus, and, then, how we leverage NFN.

5.4.1 Factor reduction

Let $\phi(X)$ be a $factor$, and an *evidence* $E = e$. We define the reduction of the $factor$ ϕ to the context $E = e$, denoted $\phi[E = e]$, to be a $factor$ over scope $X - E$, such that: $\phi'[E = e](X - E) = \phi(X - E, E = e)$ [82].

To this end, we need to consider how we introduce the *evidence*. The *factor reduction* function is built on the idea of erasing invalid entries that conflict with the introduced *evidence*. For example, given the evidence $X_e = x_e$, and a $factor$

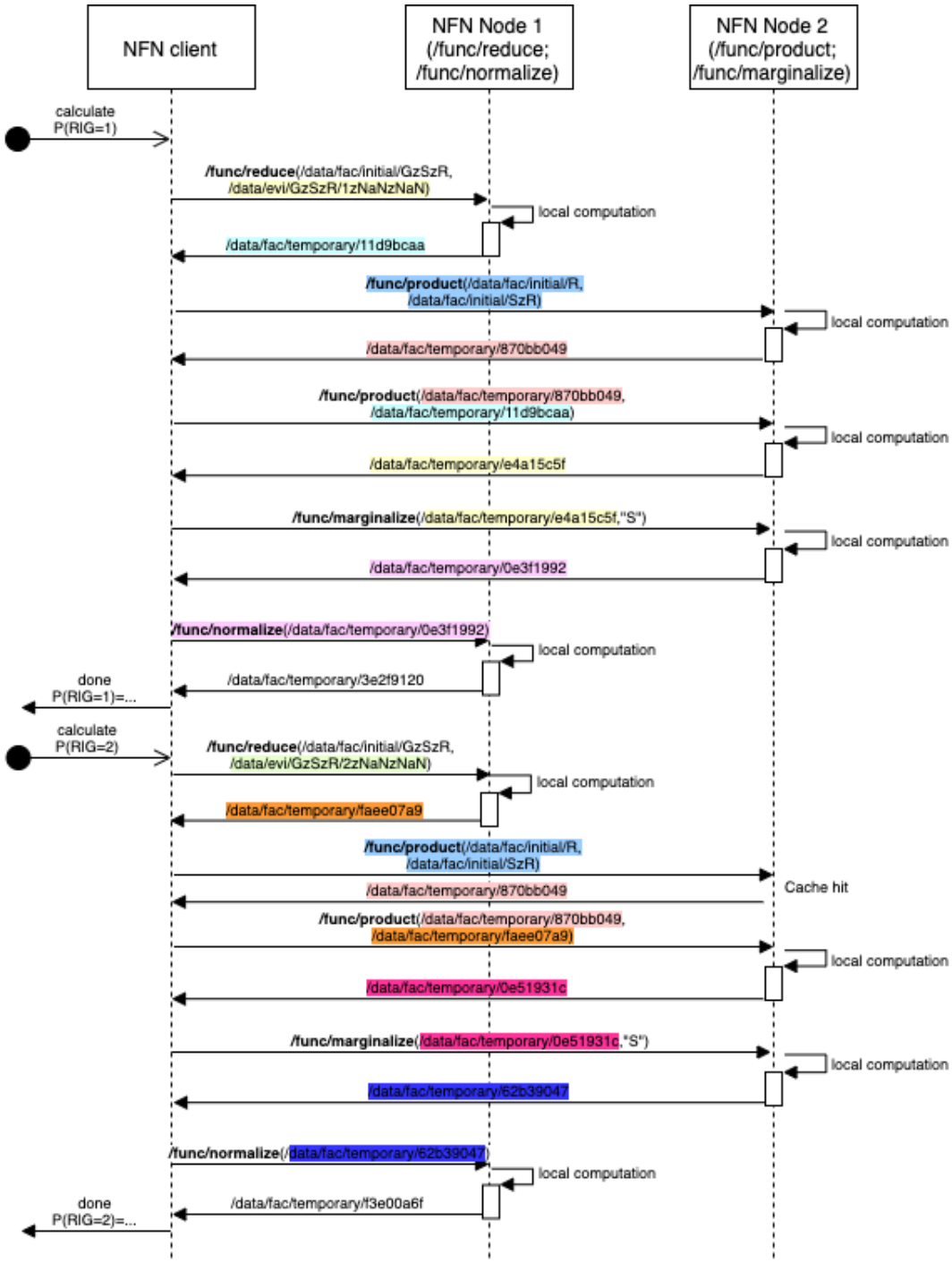


FIGURE 5.5: Sequence diagram of calculation P(R | G=1) and P(R | G=2)

$\phi(X_1 \ldots, X_n)$, the new *factor* $\phi(X_1, \ldots X_{e-1}, X_e = x_e, X_{e+1}, \ldots, X_n)$ is constructed by removing entries which include $X_e \neq x_e$.

The value of the *evidence* in our approach is designed to mark that the entry should be dropped in the *factor*. To identify the entry which should be dropped, we use the value 0, and for the entry that should be retained after the operation, value 1. As shown in figure 5.3, the third part of the *factor* is a serialization of the values of

TABLE 5.1: List of helper functions

Function	Description
serialize	Serializes an array of *factor*'s values to a string
deserialize	Deserializes a string to the array of *factor*'s values
productEviVal	Multiplies values of *evidence* of two variables
verifyValidIndex	Verifies the index of values of product of two *factors* is valid
mergeVarDim	Merges the set of random variables and their dimensions of two *factors*
removeVar	Removes a variable from the set of random variables in case of factor reduce or factor marginalization
getVarDimFac	Retrieves the set of random variables and their dimensions of a *factor* as a string
getVarDimEvi	Retrieves the set of random variables and their dimensions of *evidence* as a string
getValFac	Retrieves the values of a *factor* as a string
getValEvi	Retrieves the values of *evidence* as a string
listIdxFac	Generates the list of index for values of a *factor*

a *factor* in the right order. However, the value *evidence* is just a fusion of *evidences* from different variables. Therefore, a function is constructed that allows merging the *evidence* of variables to an intermediate data that is compatible with the dimension and the right order of the *factor* to reduce. Function *productEviVal* allows merging of *evidences* from all the random variables in the *factor* and providing new data that is compatible with the *factor*. The idea of the algorithm is that we realize the product between values in the *evidence* of two variables (line 2). In particular, if the value in one of the variables is already marked 0 to be dropped, it also means that the entry consisting of this value with all values of other variables will be marked 0, and will also be dropped.

Function *productEviVal(evi1,evi2)*
| return reduce(lambda x,y:x+y,[[i*j for i in evi1] for j in evi2])

The following algorithm enables the introduction of an *evidence* into a *factor*. Firstly, function *getVarDim* allows merging names of variables and their dimensions of *factors* and *evidences* into one, which is the meta-data of the new *factor* (line 2). Secondly, values in the *factor* and *evidence* are calculated to establish the data of the new *factor*. If the value of the *evidence* is 0, this means that the entry of the *factor* will be dropped (line 4). The result will be marked -1 and as a value of a valid *factor* is always positive, this entry will later be dropped. By contrast, if the value of the *evidence* is 1, which means that the entry is still valid, the value in the new *factor*

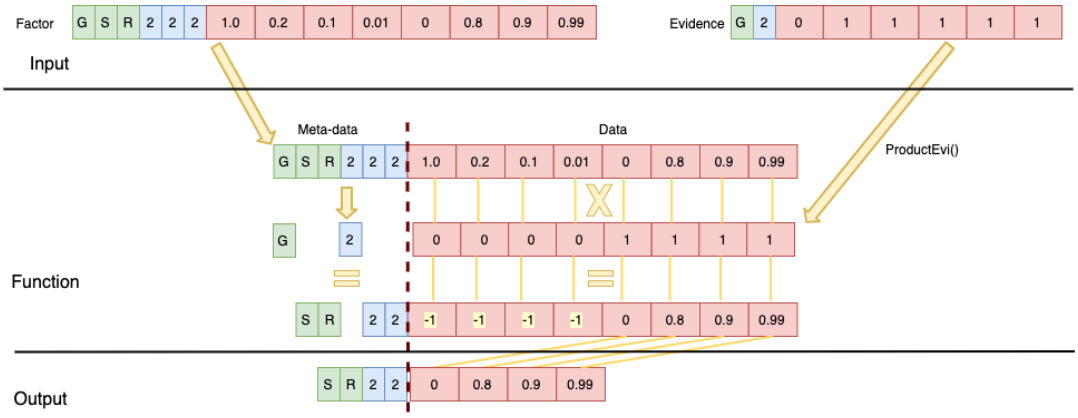


FIGURE 5.6: Example of function factor reduction (ϕ_3(G=2,R,S))

is retained by multiplying it with 1 (line 4). Finally, all of the invalid values in the *factor*, i.e. those which are negative, are removed to finally obtain a valid *factor* (line 3). Figure A.10 illustrates this process.

Function *reduceFactor(fac,evi)*

return getVarDim(fac,getVarDimEvi(evi)) + SEPARATOR_CHAR serialize(filter(lambda x:x >-1, map(lambda x,y: -1 if y == 0 else x*y, deserialize(fac), reduce(lambda x,y: productEviVal(x,y),getValEvi(evi)))))

5.4.2 Factor product

The next function that is transformed into a λ-calculus function is the *factor* product. The function is defined as follows. Let X,Y and Z be three disjoint sets of variables, and let $\phi_1(X,Y)$ and $\phi_2(Y,Z)$ be two *factors*. We define the *factor* product $\phi_1 \cdot \phi_2$ to be a *factor* ψ: $Val(X,Y,Z) \rightarrow \mathbb{R}$ as: $\psi(X,Y,Z) = \phi_1(X,Y) \cdot \phi_2(Y,Z)$ [82].

The goal of this operation is to multiply the values of two *factors* together. The joint distribution of the resulting *factor* is the merging of two scopes. However, several merging entries are not valid. For example $\phi(X=1,Y=1) \cdot \phi(Y=2,Z=1)$ is an invalid entry as $Y=1$ and $Y=2$ are in conflict and should be dropped. From this, our proposed approach consists in using the function *verifyValidIndex* to mark it with 1 if the entry is valid and 0 if it is not (line 6). Finally, we multiply the values of this function with the values of a *factor* if it is positive and mark it with -1 if it is zero (line 4). Following the same methodology with the function *factor reduction*, entries with values -1 are considered as invalid entries and are dropped in the final *factor* (line 3). Figure 5.7 illustrates this process.

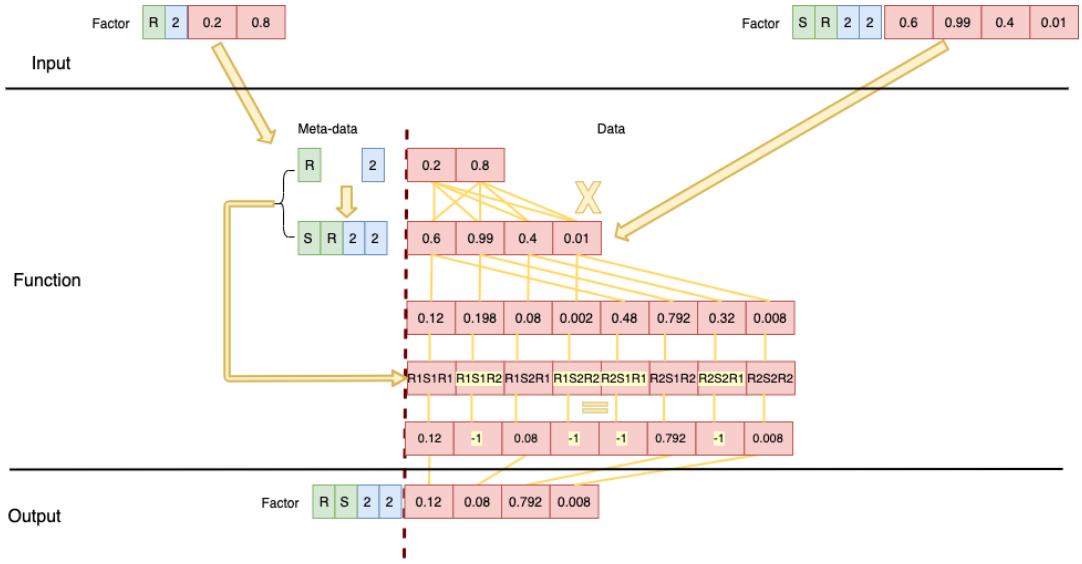


FIGURE 5.7: Example of function factor product ($\phi_1(R)\phi_2(S,R)$)

Function *productFactor(f1,f2)*

return getVarDimFac(mergeVarDim(f1,f2)) + SEPARATOR_CHAR serialize(list(filter(lambda x:x >-1, map(lambda x,y: -1 if y == 0 else x*y, productValFac(f1,f2), verifyValidIndex(f1,f2)))))

5.4.3 Factor marginalization

To achieve the sum-product operation, a *factor marginalization* is needed. This function is defined as follows. Let X be a set of variables, and $Y \notin X$ a variable, and let $\phi(X,Y)$ be a $factor$. We define the $factor$ marginalization of Y in ϕ, denoted $\sum_Y$, to be a $factor$ ψ over X such that: $\psi(X) = \sum_Y \phi(X,Y)$ [82].

The idea of this algorithm is that we have a joint distribution and want to obtain a new distribution that eliminates a random variable. To that aim, we compute the sums in the margin over the distribution of the variable being eliminated. For example, in the case of the $factor$ illustrated in Table I, we want to eliminate variable S. We will sum-up values of $\phi_2(S=1,R=1)$ and $\phi_3''(S=2,R=1)$ to get values for the new $factor$ $\phi'''(R=1)$. To perform this, we construct the function *listIdxFac*, which lists all entries of variables in the $factor$ but not the variable to eliminate (line 5). For example, let f be the $factor$ in Table I:

listIdxFac(f,"S")=["R1","R1","R2","R2"]

Thanks to the *listIdxFac* function, we list all unique entries of random variables of the new $factor$ (line 6):

listIdxFac(getVarDimFac(removeVar(f,"S")) ,"S")=["R1","R2"]

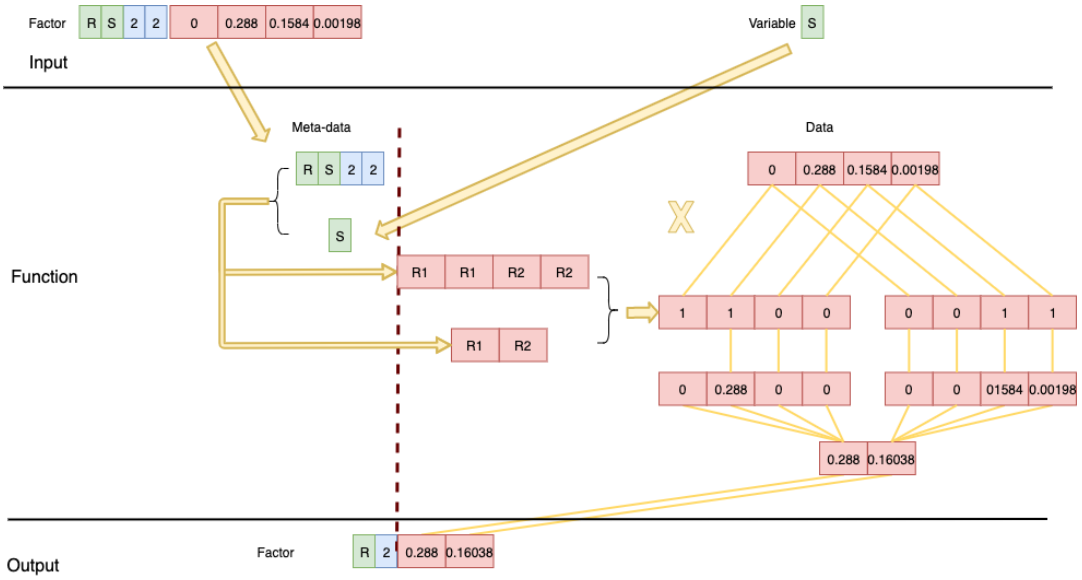


FIGURE 5.8: Example of function factor marginalization $(\sum_S \phi''(R,S))$

Then, for each value in the list of unique entries, we locate the positions of duplicated entries and mark them with 1 (line 5). This step allows us to know the positions of values that belong to the same entry in the new *factor*, and sum up these values. Finally, the value of ϕ in the new *factor* is calculated from the sum of duplicated entries where their positions are marked as 1. Figure 5.8 illustrates this process.

Function *marginalizeFactor(f,v)*

> return getVarDimFac(removeVar(f,v)) + SEPARATOR_CHAR serialize(map(lambda z:reduce(lambda x,y:x+y, map(lambda x,y:x*y, getValFac(f),map(lambda x:1 if z in x else 0,listIdxFac(f,v)))) ,listIdxFac(getVarDimFac(removeVar(f,v)),v)))

5.4.4 Factor Normalization

Factor normalization is the simplest of all the functions described. Here, we divide all the values of the *factor* (line 5-6) by their sum to normalize the *factor* (line 3) and finally obtain a conditional probability: $P(X_q|E = e) = \frac{1}{Z}\prod_{i=1}^{n}\phi'_i(X_i)$ [82] where $Z = \sum_{i=1}^{n}\phi'_i(X_i)$. Figure 5.9 illustrates this process.

Function *normalizeFactor(fac)*

> return getVarDimFac(fac) +SEPARATOR_CHAR serialize(map(lambda x,y:x/y, deserialize(getValFac(fac)), [reduce(lambda x ,y : x+y , deserialize(getValFac(fac)))] *len(list(deserialize(getValFac(fac))))))

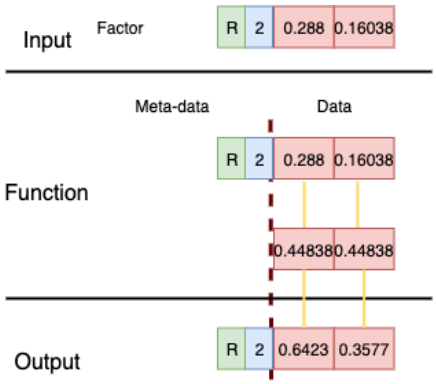


FIGURE 5.9: Example of function factor normalize ($\frac{1}{Z}\phi'''(R)$)

5.5 The use case: the Content Poisoning Attack

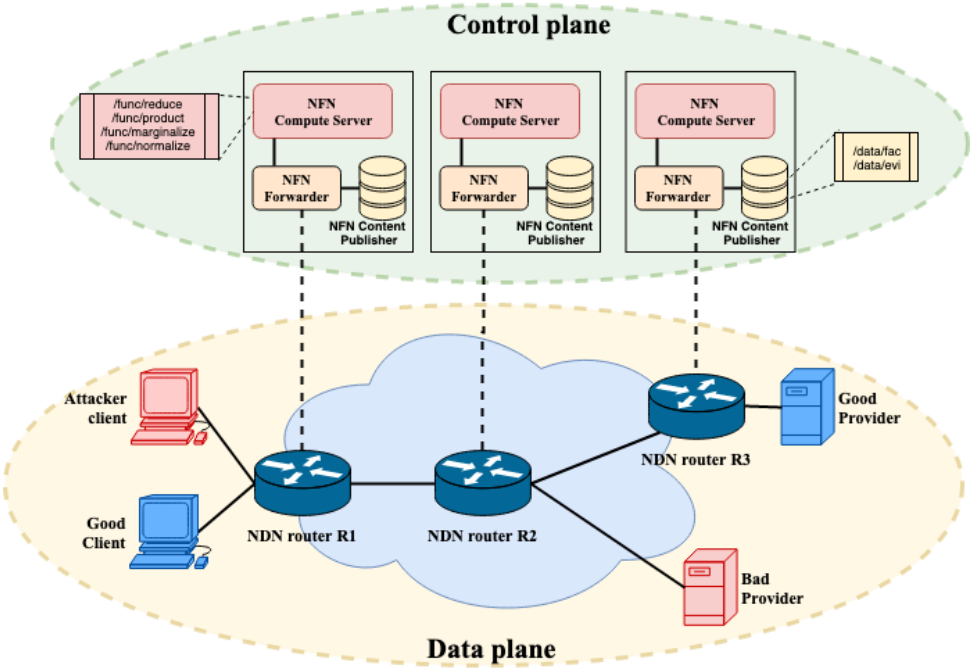


FIGURE 5.10: Experiment topology

To evaluate the performance of our approach, the Content Poisoning Attack (CPA) use-case is still considered. The topology already considered in chapter 3, containing three NDN routers with both NFN forwarder and a NFN compute server, is re-used, as depicted in Figure 5.10. Each node integrates all functions, data, and exchange results. As an anomaly detection function to compute with NFN, the BN we have previously exposed is considered. We remind here that it consists of 19 discrete nodes: an Anomaly node and 18 different metrics. MMT monitoring probe are still coupled with each router to extract and collect data concerning the 18 metrics every 5 seconds. However, to demonstrate and evaluate all functions in the VE algorithm, we consider one of the metrics as unknown; otherwise, the *factor marginalization* function would not be used as there would be no variable to eliminate.

The proposed approach is implemented using PiCN [152], which is the newest

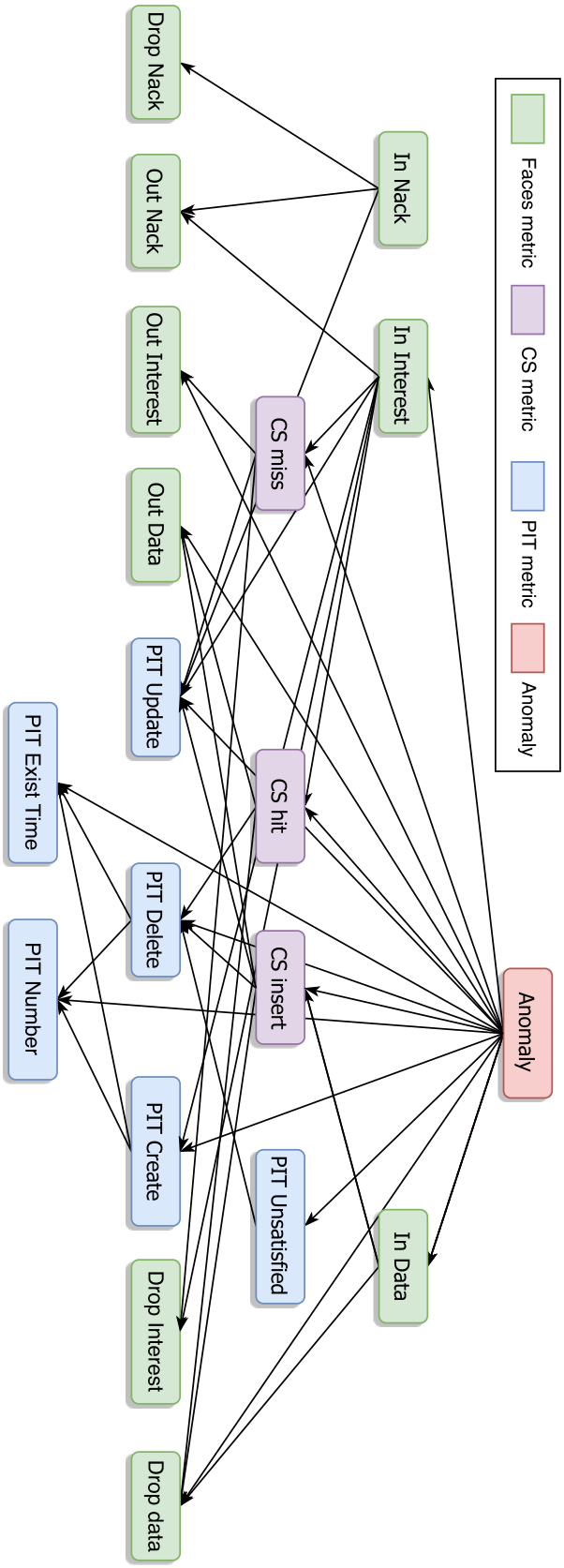


FIGURE 5.11: BNC as CPA detection

version of NFN, written in python. It is compared with a standard Bayesian Network inference algorithm leveraging an open-source library named pgmpy [153] also written in python.

5.6 Evaluation

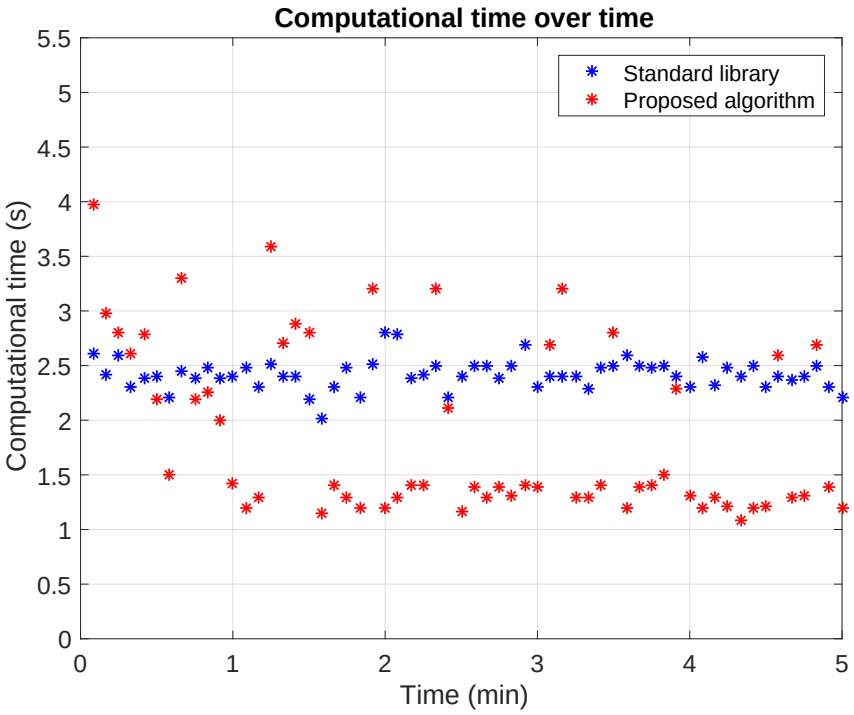


FIGURE 5.12: Snapshot of computational time over time;

We evaluate the performance under different conditions to determine how the anomaly detection can be improved when using NFN.

5.6.1 Performance over time

To assess the performance of our approach, we simulate normal traffic during 10-minute intervals and measure the evolution of the processing time, cache hits, local computation and requests to other nodes. As expected, at the beginning the cache is almost empty as shown in figure 5.13. Hence, the computational time of the proposed approach is higher than what it will be later as shown in figure 5.12. This explains the need for higher processing time at the beginning when compared with a standard non-optimised approach. The effectiveness of our solution is shown when the cache is filled with results from previous executions or from results calculated by neighboring nodes. In this case, the computational time is lower than that of the standard approach. This is confirmed by the result of Figure 5.13, which shows

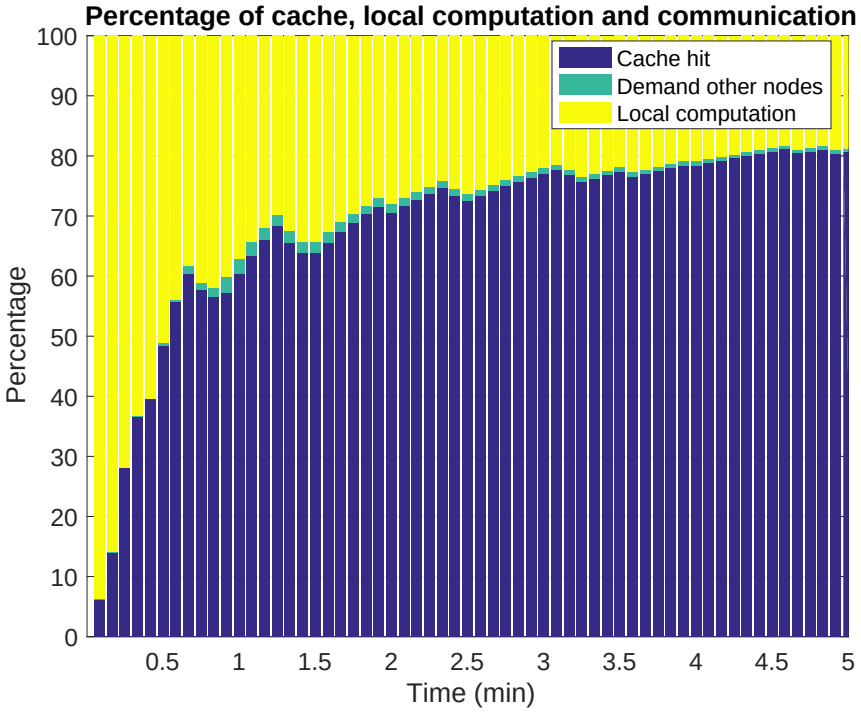


FIGURE 5.13: Evolution of proportion of requests over time

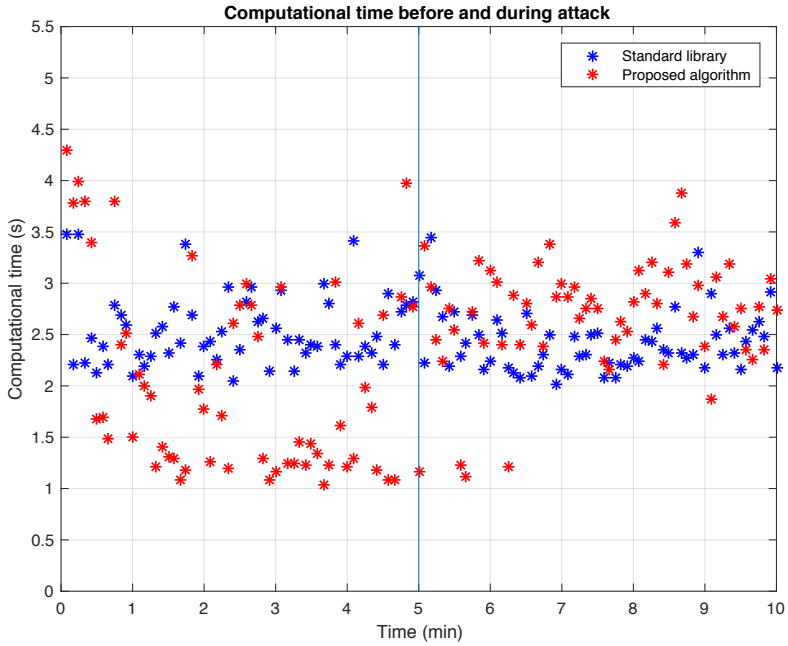


FIGURE 5.14: Snapshot of computational time before and during attack

that the average percentage of local computation and requests to other nodes are the highest at the beginning. Furthermore, we note that the portion of the cache hits

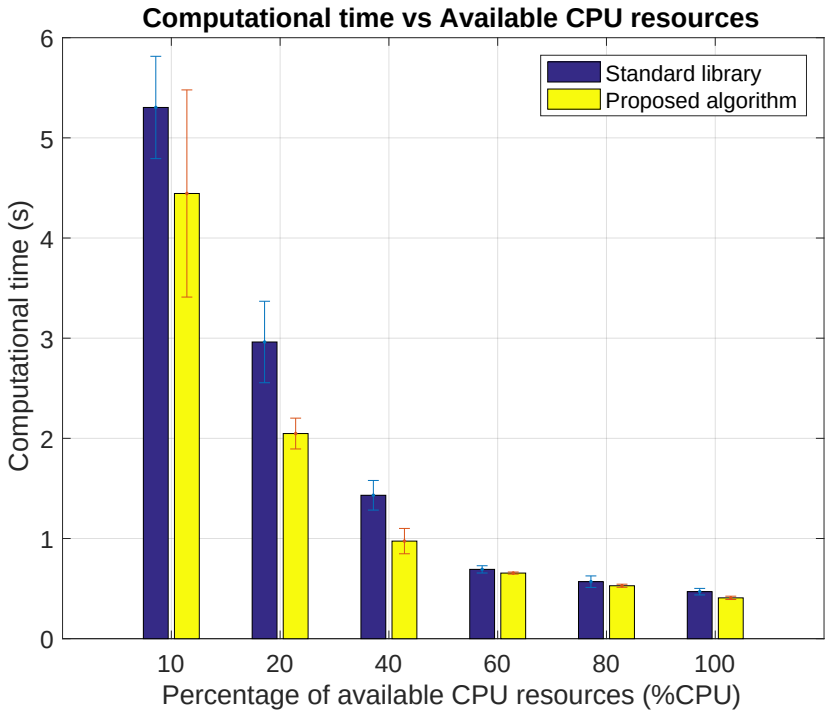


FIGURE 5.15: Impacts on computational time: Available CPU resource

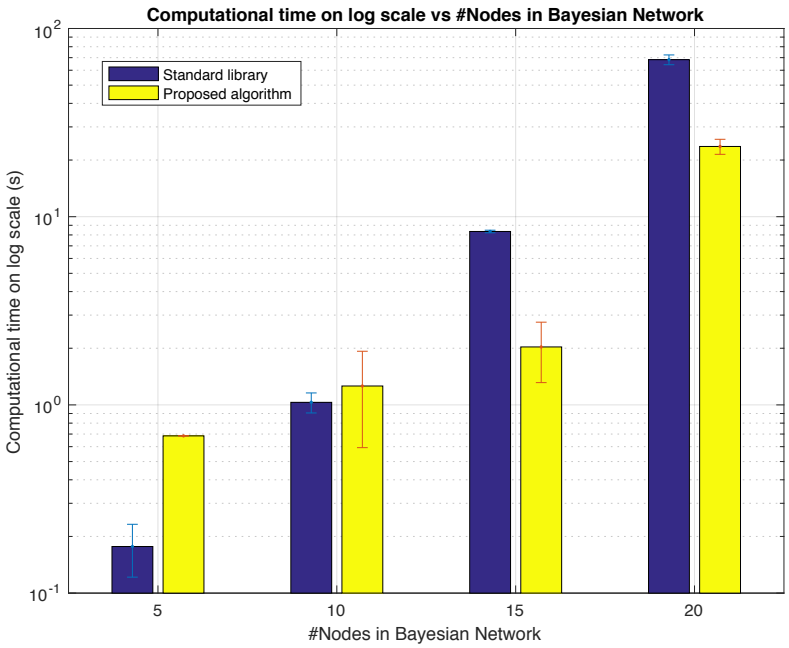


FIGURE 5.16: Number of nodes in BN

initially increases over time, then stabilizes since the cache reaches its limit. The percentage of the cache hits after 10 minutes is 80%. This shows the effectiveness of our approach.

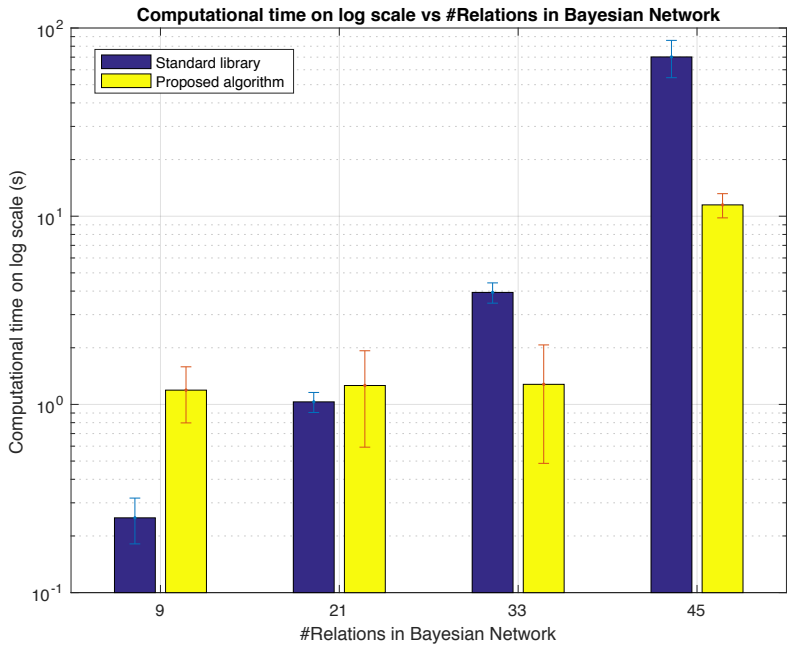


FIGURE 5.17: Number of relations in BN

5.6.2 Impact of normal and abnormal traffic

Another evolution of the computational time, illustrated in Figure 5.14, is evaluated in the case of an attack, where we simulate normal traffic during 5 minutes, and then the CPA attack during 5 minutes. The vertical blue line marks the time at which the attack starts. As we can see, after the attack the computational time increases. Since the metrics during an attack are abnormal in comparison to normal traffic, the detection function cannot be found in the cache and it requires additional time to compute the operations in the inference algorithm. To conclude, NFN is efficient in normal traffic but, when the attack occurs, the proportion of repeated computational operations decreases, so the computation time using NFN increases.

5.6.3 Impact of available CPU resources

The result shown in Figure 5.15 proves that the computational time in our approach is lower than that of the standard approach when the computation capacity of the hosting node is limited. When the available CPU resource of the router is smaller than 0.4, the proposed approach performs better. The reason is that when the router has sufficient computational resources, it will consume all of the resources to execute the operations of the inference algorithm. Therefore, a standard approach can consume all the resources and perform computational operation rapidly. On the other hand, when it reaches its computation limit (in case of limited resources), the use of

the cache helps to better execute the computational operations than with the standard approach. It reduces the computational time by 10% to 30%. Figure 5.18 also illustrates the usage of CPU during these experiments. We can see from Figure 5.15 and 5.18 that we can benefit from NFN when the computational capacity is limited, which is the case in, e.g., IoT-type networks.

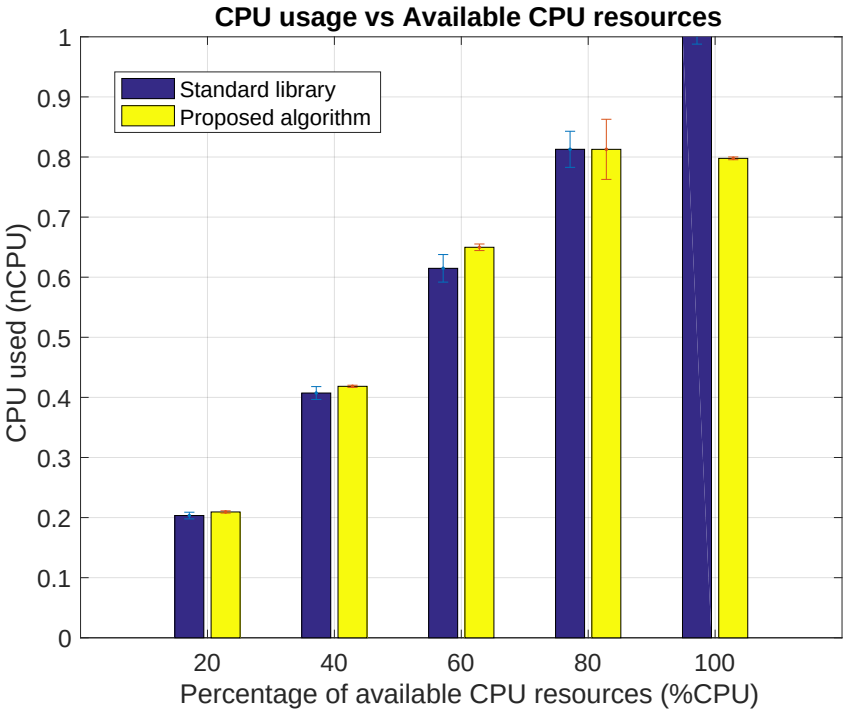


FIGURE 5.18: Impact of available CPU resources on CPU usage

5.6.4 Impact of the BN complexity

To evaluate the performance of our approach according to the complexity of BN, the number of nodes in our BN and the number of relations is considered. As shown in Figure 5.16 and 5.17, when the BN is simple (with a small number of nodes, less than 10, or relations, less than 21), the computational time when using the standard approach is undoubtedly better than in our approach. The reason is that the number of computational operations is ordinary, which means that it does not need the help of a cache or other nodes to perform the computations. However, when the BN becomes complex, the performance of our approach is substantially better than the standard method. In this case, our approach calculates 4 to 8 times faster than the standard one. In fact, in this case, the number of computation operations is huge and exceeds the computational capacity of the node. As a consequence, the use of a cache and the help of other nodes become extremely important. The results show

that we benefit from the NFN infrastructure when the BN is complex and needs a significant amount of operations.

5.6.5 Impact of latency between nodes

The impact of latency between nodes also needs to be considered. One can note that a drawback of the proposed architecture is its dependency on the latency, as when the latency increases, the computational time increases as well because the latency presumably has an impact on the communications between the nodes to execute the computational operations. To show this, we perform experiments to evaluate the impact of the latency on the performance of the proposed algorithm, as shown in figure 5.19. We can see that when this latency is too high (greater than 100ms), the computational time does not noticeably increase. By contrast, it is smaller than in the case where the latency is 20ms. The latency has only a limited impact on the computational time when we change the latency from 0 to 20ms. The reason for this phenomenon is that when the latency is high, i.e. at the beginning, the NFN router sends requests to neighbors but does not receive any results. When the timeout is triggered several times, the router does not send requests to the other nodes, and considers these routes as unreachable, and then decides to calculate the operations locally. As a consequence, in the case of 50ms or 100ms of delay, the operations are executed faster. While in the case of 20ms of delay, the routers wait for the response from the neighbors because the *Interests*' lifetime has not yet expired. In brief, the latency has an impact on the proposed architecture. However, a larger delay does not mean lower performance. The proposed algorithm gets the best performance when the delay is close to zero. The next cases that get good performance are when the delay is significant, as in that case, the node decides to execute the computation locally. Meanwhile, the worst performance happens when the delay is not close to zero but not large enough to guide the node to execute the computation locally, then it is impacted by the network's latency.

5.7 Conclusion

Anomaly detection is a critical and complex task which can be optimised by using NFN. In this chapter, we have defined the core elements for the design and implementation of an NFN-supported Bayesian Network inference algorithm. To this aim, we have shown that the VE algorithm in BN can be transformed to λ-calculus functions and then, thanks to NFN, it is possible to cache and share results between nodes. In the context of Content Poisoning Attack detection, we have demonstrated

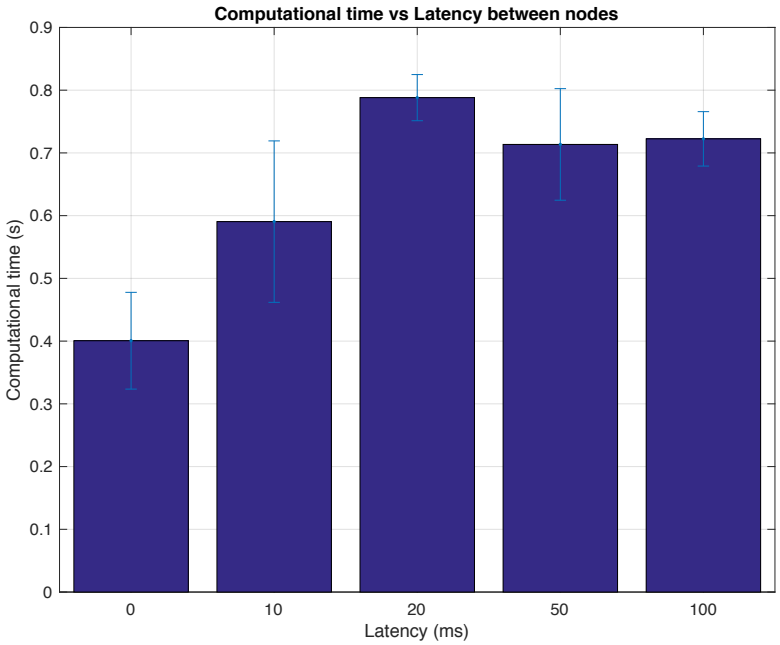


FIGURE 5.19: Impact of delay on computational time

that an NFN-supported BN performs better not only in the case of limited computational resources but also when the BN is complex, thus proving the benefit of this approach for complex anomaly detection functions operated in the context of IoT, for instance.

Our work is the first brick to build a content-oriented control plane. Indeed, the current approach could be extended to integrate other methods for anomaly detection. Moreover, other attacks could also be considered to demonstrate the applicability and generality of the proposed approach.

Conclusions and Perspectives

Conclusions

Nowadays, the Internet has evolved far beyond its initial design. In the era of social networking and content sharing, the legacy host-centric design of the Internet is struggling to resolve emerging problems such as security, management, and quality of service. In the research communities, various approaches such as Multicast, CDN, P2P are proposed to fill the gap between the current usage of the Internet and its initial design. However, these solutions are still based on current Internet design and cannot solve the problem thoroughly.
The Information-Centric Networking paradigm is proposed as a clean-slate approach for the Future Internet to address in an integrated way all known issues. Named Data Networking is the most mature proposal among existing ICN architectures. A novel approach also requires a significant effort to deploy, manage, and secure it. As a result, solutions to deploy securely and effectively NDN are crucial for network operators before deploying NDN on a large scale. In this thesis we have proposed a solution to this requirement through a novel content-oriented security framework for NDN.

Before deploying NDN, all potential threats must be identified and detected. Among identified attacks in NDN, CPA is the one that got the most attention from the NDN research community as it is an NDN-specific attack and is hard to be detected by any detector based on a single metric. As a result, we carefully considered different approaches to detecting such an attack that impacts various metrics like CPA. We have shown that the Bayesian Network Classifier fits our demand for a network anomaly detector in NDN as it can correlate multiples events and their impacts on a set of metrics to classify between normal and abnormal behaviors. Besides, BNC can also naturally handle the underlying random nature of observed metrics using Bayes probabilistic approach hence able to handle the dependencies between variables and events relevant in an attack. However, like other machine learning or probabilistic methods, BNC is costly in terms of resource consumption and computational time.

We tackled the first limitation by proposing a security monitoring plane for

NDN as it is indispensable before any potential deployment of this novel architecture in an operating context by any provider. A metrics list is proposed to cover the data plane and feed a detector to tackle not only current attacks in NDN but also possible not yet identified future ones. However, two questions arose for the proposed monitoring plane. Firstly, the proposed metrics represent various characteristics of different NDN components. For example, the number of entities in *PIT*, the number of Cache hit in *CS*, or the average time of expiration in *PIT*. The values of these metrics vary in different ranges. As a consequence, the abnormal behavior of different metrics cannot be detected by a unique detector. Secondly, anomaly behaviors and attacks could impact several metrics, but some changes in the network load could also change various metrics. Two layers of detection are proposed respectively to address these issues. Firstly, we have designed a micro detector to detect if a metric is shifted from its normal behaviors; the micro detector depends on only the behavior of a metric. To overcome the second issue, we proposed a correlation engine using the Bayesian Network that correlates the alerts from all metrics and performs the inference to predict if a router is under attacked. Using CPA as the use-case, we collected data from a real NDN testbed and assessed our proposed monitoring plane. The numerical results prove that even for a small attack payload, the monitoring plane can detect a CPA with high efficiency and precision.

Deploying NDN is a strategic and complex task which can be supported by a combination of SDN/NFV and smart orchestration. To this aim, we propose a secure content-oriented orchestration. We have defined the core elements for designing and implementing a content-based orchestration for virtualized NDN networks. We have shown that the TOSCA standard can be extended to become a solid content-based service specification template. With novel orchestration components, we have demonstrated that a content-oriented orchestrator can both automatically deploy a virtual NDN topology and enforce dynamic reconfiguration in the virtual network triggered by security and scale-out policies. To evaluate the proposed orchestrator, we collected data from different evaluation experiments. The numerical results have demonstrated that the proposed orchestrator can both automatically deploy a virtual NDN topology, but also enforce dynamic reconfiguration in the virtual network triggered by security and scale-out policies.

Finally, we come up with the proposal of a content-oriented anomaly detection architecture. The proposed content-oriented anomaly detection architecture is motivated by our observation that a security plane continuously monitors a large-scale network; a large number of security operations are repeated. We have defined the core elements for the design and implementation of an NFN-supported Bayesian Network inference algorithm. To that aim, we have shown that the VE algorithm

in BN can be transformed to λ-calculus functions and then, thanks to NFN, cache and share results between nodes. In the context of CPA detection, we collected the numerical results to demonstrate the performance of the proposed algorithm. The numerical results prove that the proposed NFN-support BN performs better than standard BN in the case that the BN structure is complex and in the case of limited computational resource. As a result, the proposed NFN-support could prove its benefit for complex anomaly detection functions operated in limited computational resource devices.

Perspectives

The work achieved in this thesis opens several perspectives. In the initial context of this thesis, the proposed monitoring plane could still be improved. Moreover, in the long term, the work accomplished in this thesis opens the way towards a content-oriented control plane that can be extended to be used as an application for anomaly detection in general.

As NDN is still a novel architecture and is still in an academic and experimental environment, potential attacks could emerge when the architecture becomes more popular. In our proposed solutions, a list of metrics is proposed to cover all elements in the NDN data planes. As a consequence, whether it is a novel attack, the proposed monitoring plane could also monitor its behaviors. Then, its behaviors will be used to train the Bayesian Network, so the proposed monitoring plane is extensible for other NDN attacks. However, the proposed BNC is dependent on known attack's behaviors. As a result, it takes time to train the anomaly detector when a new attack emerges. The training phase is a drawback of our proposed monitoring plane as a "zero-day" attack could exploit it. More specifically, each known attack is currently represented as a class in the output node of the BNC among the normal class representing the normal traffic. To overcome the issue mentioned above, we can add another BNC with the same structure, but the output node consists only of two classes: abnormal and normal to alert if any abnormal behaviors have occurred in the network without identifies the type of attack. The data collected from all known attacks could be classified as abnormal, while the data collected from normal traffic or a legitimate change in traffic load is classified as normal. Regarding Bayesian Inference, the work in this thesis only considers the VE algorithm as an Inference algorithm. Nevertheless, other inference algorithms can also transform into λ-calculus to leverage the benefit of the cache in NFN. The Clique tree algorithm is such an example, as its nature is proposed to cache the computation during the inference process.

The initial goal of the proposed approach is to secure the deployment of NDN. However, the hypothesis that security functions repeat their operations does not limit only to the context of BNC and NDN. For example, regarding the scenario, we can consider the case of a network traffic anomaly detector. The traffic of a server in off-peak hours unquestionably does not change much, and hence their behaviors are repeatedly over the off-peak hours. Another example is the case of intrusion detection camera-based. In this case, if there is no suspicious behavior, the output of the camera is unchanged. In both cases, as the input for the security functions does not change, we can benefit from the cache. Regarding the algorithm for security functions, a good example is deep learning, where the detection is based on multi-layer neural networks. As a result, the computation for these different layers could be distributed and cache on the network. Moreover, the work accomplished in this thesis is designed and developed in a real deployment, therefore opens the way towards a content-oriented control plane can be used as an application for another scenario/environment. This capacity opens a new door for the proposed approach to be used in NDN and in other network architecture, especially whenever the gap between the security functions and the constraint resources of execution nodes is significant.

Appendix A

Résumé de la thèse en français

Contents

A.1 Introduction

A.1.1 Contexte

Internet a été initialement conçu pour permettre à ses usagers de se connecter à une machine distante, accédant ainsi à des ressources ou à un échange digital avec d'autres usagers. Tout au long de sa croissance exponentielle, il s'est transformé pour devenir une infrastructure utilisée majoritairement pour la distribution massive de contenus. En effet, l'essor des services et usages mobiles, l'Internet des objets, les réseaux sociaux et le streaming vidéo ont littérallement fait exploser la quantité de données circulant sur les réseaux. Un pourcentage notable de ces données est aujourd'hui dupliqué étant donné que les contenus audios, les vidéos et les données volumineuses sont distribués par un faible nombre de fournisseurs de services à de très nombreux utilisateurs finaux. Bien que plusieurs solutions aient été proposées au niveau des différentes couches protocolaires et des services associés, tels que le multicast, les réseaux de distribution de contenu (CDN) ou les solutions pair-à-pair (P2P), l'architecture actuelle d'Internet n'a pas été conçue pour permettre directement la distribution de contenus à grande échelle et à grand nombre d'utilisateurs. C'est pour cela que de nouveaux paradigmes sont apparus. Parmi eux, le paradigme des réseaux orientés information (en anglais Information-Centric Networking (ICN)) a été proposé pour passer du paradigme traditionnel de communication d'hôte-à-hôte de l'Internet actuel à celui d'hôte-à-contenu. Parmi toutes les approches ICN existantes, Named Data Networking (NDN) est la plus mature et la plus adoptée. Toutefois, face à l'enracinement fort du protocole IP, des leviers de déploiement sont apparus comme nécessaires pour permettre aux solutions ICN d'être réellement mises en production.

De plus, comme la virtualisation des serveurs s'est imposée dans les centres de données depuis une vingtaine d'années, les contraintes des équipements matériels ont conduit les opérateurs de télécommunication à appliquer ces dernirèes années des technologies de virtualisation à leurs fonctions réseau. Cette nouvelle approche des systèmes communicants est nommée virtulialisation de fonctions réseaux (en anglais Network Functions Virtualization (NFV)). La virtualisation des fonctions réseau sur des pools partagés de ressources matérielles standardisées permet d'atteindre

une plus grande agilité, flexibilité et élasticité dans la configuration et le fonctionnement du réseau, conduisant à une réduction du temps de mise en production pour le déploiement de nouveaux services tout en réduisant les coûts d'exploitation et d'équipements. La combinaison de ces deux concepts à savoir NDN et NFV devient une évidence.

A.1.2 Problèmatique

Malgré la maturité d'ICN, les opérateurs réseau hésitent à intégrer cette nouvelle technologie dans leurs infrastructures de production. Le modèle de la virtualisation des fonctions réseau (NFV) permet aux opérateurs de réseau de déployer des fonctions de réseau dans des logiciels qui fonctionnent sur un matériel de serveur standard, couplé avec des services de réseau définis par logiciel (SDN), un excellent candidat pour prendre en charge le déploiement de réseaux ICN. Bien que les principaux blocs fonctionnels de gestion et d'orchestration (MANO) soient désormais normalisés par l'ETSI et mis en œuvre par des implementations NFV, ils visent principalement des fonctions de réseau virtuel (VNF) basées sur IP. Or, la mise en oeuvre de VNF non IP, tels que NDN, présente un impact sur le plan de gestion et ses différentes fonctions, nécessitant ainsi des évolutions du plan de management. En particulier, la sécurité des données étant indispensable, elle doit être bien conçue avant d'envisager tout déploiement de NDN dans un environnement NFV/SDN. Par conséquent, la recherche d'un orchestrateur orienté contenu sécurisé pour NDN est cruciale et constitue un défi à lever.

Avant de déployer un orchestrateur orienté contenu sécurisé, les opérateurs de réseau doivent être en mesure de surveiller et collecter efficacement les métriques (et autres indicateurs) d'un système distribué car cela permet à l'orchestrateur orienté contenu sécurisé de prendre des décisions en matière de gestion et de sécurité. A notre connaissance, il n'existe aucun mécanisme disponible dans NDN pour collecter un ensemble riche et complet de métriques. De plus, pour aider l'orchestrateur à prendre des décisions et à réagir contre les attaques, les fonctions de sécurité doivent être conçues pour corréler les métriques et sur cette base, de détecter les anomalies dans le réseau. Un tel détecteur d'anomalies fait encore défaut dans le NDN. Ces problèmes conduisent à la nécessité de concevoir un plan de surveillance de la sécurité dans NDN.

Les fonctions de sécurité, en particulier les détecteurs d'anomalies, sont des tâches gourmandes en termes de ressources de calcul et de mémoire. Comme elles sont exécutées périodiquement dans le réseau et qu'elles se basent en général sur des algorithmes statistiques ou d'apprentissage (Machine Learning en anglais), elles nécessitent un effort de calcul important. En conséquence, un défi relatif à la réduction

de l'écart entre la complexité des fonctions de sécurité et les ressources disponibles dans des nœuds NDN, existe et doit être relevé. Ainsi, une architecture de détection d'anomalies distribuée est nécessaire.

Pour résumer, le déploiement sécurisé de NDN par les opérateurs de télécommunication nécessite une recherche approfondie sur un plan de sécurité orienté contenu pour NDN. Ce plan de sécurité proposé doit inclure:

- Un plan de surveillance de la sécurité pour collecter différentes métriques et détecter différentes attaques et anomalies;
- Un orchestrateur orienté contenu sécurisé pour déployer, gérer et sécuriser les fonctions du réseau NDN;
- Une architecture de détection d'anomalies distribuée et performante.

Ces thématiques seront abordés dans cette thèse.

A.1.3 Contributions

L'objectif de cette thèse est de proposer un plan de sécurité orienté contenu pour NDN qui puisse traiter efficacement les menaces de sécurité relatives à NDN, tel que l'attaque d'empoisonnement de contenu (Content Poisoning Attack (CPA) en anglais). Dans un premier temps, nous concevons et mettons en œuvre un plan de surveillance pour NDN. Dans ce but, une liste de métriques a été construite et sélectionnée et, pour chacune d'elles, des micro-détecteurs, capables de prendre en compte toute variation anormale, ont été conçus et évalués à la fois théoriquement et empiriquement. Les corrélations entre les métriques ont été décrites pour détecter tout comportement anormal dans un nœud NDN en exploitant un classificateur de réseau bayésien (BNC). L'attaque de type CPA est notre choix pour valider notre approche proposée. Les résultats numériques obtenus dans des expériences de déploiement réel démontrent l'efficacité du plan de surveillance, en particulier le détecteur d'anomalies proposé. Cependant, nous avons constaté que certaines améliorations sont nécessaires pour assurer une bonne performance au détecteur d'anomalie. Ces améliorations sontabordées dans la troisième partie de cette thèse.

Deuxièmement, l'étape suivante de la détection d'attaque est sans aucun doute les atténuations et les réactions. Pour cela, une orchestration orientée contenu est proposée pour déployer un réseau virtuel NDN sécurisé. La norme TOSCA est exploitée à l'aide d'une extension orientée NDN spécialement conçue pour permettre la spécification des exigences de déploiement et de comportement opérationnel des services NDN. Nous mettons également en évidence les politiques de sécurité et de

performance liées au NDN pour produire des contre-mesures contre les anomalies pouvant provenir d'attaques ou d'incidents de performance.

Dans le contexte de la sécurité des systèmes distribués, nous avons en effet découvert que les solutions actuelles centrées sur les nœuds induisent un gaspillage de ressources informatiques, ce qui élargit encore l'écart entre la complexité des fonctions de sécurité et les ressources de contrainte des nœuds. De plus, nous avons également découvert que le résultat des fonctions de sécurité se répète dans le temps et que ces fonctions pourraient également être distribuées pour accélérer le temps de calcul. D'un autre côté, le concept du réseau orienté contenu correspond naturellement aux exigences pour s'attaquer à ces problèmes. En effet, nous pouvons bénéficier du concept de cache et tirer parti du réseau orienté contenu comme environnement d'exécution pour prendre en charge efficacement les fonctions de détection d'anomalies. Par conséquent, nous proposons un plan de sécurité orienté contenu pour surveiller NDN. Plus spécifiquement, les fonctions de sécurité peuvent être distribuées dans Named Function Networking (NFN). NFN est un réseau orienté contenu, qui permet de répartir la tâche de calcul dans le réseau. Nous utilisons la détection d'anomalies par réseau bayésien comme cas d'utilisation pour démontrer que l'approche proposée peut améliorer les performances, surtout si la fonction de sécurité est compliquée et les ressources de calcul limitées.

A.1.4 Organisation

Le reste de ce chapitre est organisé comme suit. La section A.2 fournit un aperçu des architectures ICN. Premièrement, nous présentons le concept d'ICN et nous nous concentrons uniquement sur la solution NDN. Enfin, nous discutons du choix de CPA comme cas d'usage de la sécurité pour notre étude. La section A.3 expose les concepts essentiels des réseaux bayésiens et en particulier l'inférence bayésienne qui est au cœur de notre solution de détection d'anomalie. La section A.4 présente notre conception de plan de surveillance pour NDN, étape par étape. Tout d'abord, nous présentons l'architecture globale du plan de surveillance. Ensuite, nous présentons des composants dédiés à NDN au coeur de notre plan de surveillance que sont la sonde de surveillance et le détecteur d'anomalie. Enfin, nous évaluons la performance globale du plan de surveillance à l'aide du cas d'usage CPA. La section A.5 se concentre sur la conception d'une solution d'orchestration pour la sécurité des réseaux orientés contenu. Nous présentons l'architecture de notre architecture NFV MANO ainsi que ses différents composants. Ensuite, nous présentons les différents résultats obtenus à partir de différentes expériences que nous avons effectuées pour évaluer les performances de notre solution d'orchestration, en particulier pour contrer les attaques CPA. Enfin, la section A.6 propose de revisiter le plan de surveillance pour utiliser une nouvelle approche orientée contenu, appelée NFN,

comme support de l'inférence bayésienne. Tout d'abord, le schéma de nommage et la structure des données sont proposés. Ensuite, nous choisissons l'algorithme le plus basique et standard pour l'inférence bayésienne exacte afin d'illustrer comment nous transformons les fonctions orginielles vers NFN. La performance de cette solution est ensuite présentée pour démontrer l'avantage et le potentiel de l'approche proposée vis à vis d'une solution d'inférence standard.

A.2 Named Data Networking - Etat de l'art

Dans cette section, nous effectuons un état de l'art sur les ICN et la solution NDN en particulier.

A.2.1 Information Centric Networking

Information-Centric Networking [7] (ICN) est une architecture réseau permettant un passage du paradigme traditionnel de communication d'hôte à hôte de l'Internet originel à un paradigme d'hôte-à-contenu. ICN identifie chaque contenu par un nom unique au lieu d'utiliser une adresse IP et met également en cache le contenu sur le réseau (sur les noeuds intermédiaires) pour un accès plus direct et moins contraignant entre le fournisseur et les consommateurs.

A.2.2 Named Data Networking

Parmi toutes les approches ICN, NDN (Named Data Networking) [9], qui suit son prédécesseur Content Centric Networking (CCN), est la plus mature et adoptée.

Nommage

Afin de permettre aux fournisseurs et consommateurs d'échanger des contenus, ces derniers doivent convenir d'une convention de nommage. Dans ce but, NDN propose un schéma de nommage lisible et structuré qui exploite un schéma hiérarchique, similaire à celui des URL : dans NDN, chaque nom de contenu contient divers composants séparés par le caractère "barre oblique". L'organisation hiérarchique dans NDN permet non seulement d'offrir un schéma de nommage lisible mais aussi de faciliter le routage, car à chaque saut dans un chemin peut correspondre un niveau dans le nom hiérarchique.

La première partie du schéma de nommage contient un nom global unique et la seconde partie permet la gestion de version des contenus. En outre, certains contenus peuvent être trop volumineux ; ils doivent être segmentés en paquets plus petits. Ainsi, la dernière partie d'un nom est réservée à la segmentation. La figure suivante illustre un exemple de nom de contenu NDN.

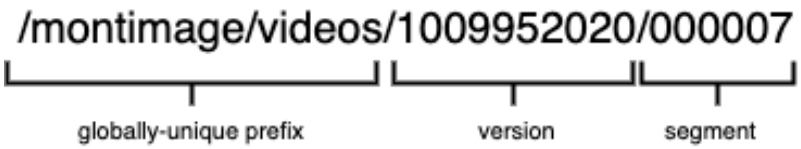


FIGURE A.1: Un exemple de nom de contenu dans NDN

Paquet

L'architecture NDN utilise un mécanisme basé sur l'extraction, utilisant deux types de paquets, *Interest* et *Data*, qui représentent respectivement une demande de contenu et un paquet de réponse. Un paquet *Interest* se compose du nom du contenu demandé, d'une *nonce*, qui agit comme un identifiant unique, et de certains guides (la portée, la durée de vie du *Interest*). Un paquet *Data* représente une réponse à un *Interest*, qui contient l'objet d'information demandé.

Un nœud NDN contient quatre composants : une *Forwarding Information Base (FIB)*, une *Pending Interest Table (PIT)*, une *Content Store (CS)* et des *Faces*.

- *Forwarding Information Base (FIB)* : conserve les informations de routage à utiliser pour acheminer les paquets *Interests* dans une liste d'entrées. Chaque entrée se compose d'un préfixe de nom et d'une liste de *Faces* sortantes qui peuvent être utilisées pour envoyer les paquets *Interests* aux fournisseurs de contenu.

- *Pending Interest Table (PIT)*: la *PIT* conserve tous les paquets *Interest* en attente d'un paquet *Data*. Chaque entrée de la *PIT* contient un préfixe de nom et la liste des *Faces* entrantes des *Interests* qui sont en attente pour ce préfixe. Ces informations sont utilisées pour renvoyer les paquets *Data* aux consommateurs sur le chemin inverse de celui emprunté par les paquets *Interests*. L'entrée est supprimée lorsque le paquet *Data* correspondant a été reçu ou lorsqu'un délai d'attente expire.

- *Content Store (CS)*: le *CS* est utilisé pour la mise en cache des paquets *Data* sur leur chemin vers leur consommateur pour améliorer les performances de la fourniture de contenus dans NDN. Chaque entrée d'un *Content Store (CS)* contient pour chaque un paquet *Data*, l'indicateur indiquant si le paquet est non sollicité et l'horodatage auquel les données mises en cache deviennent périmées.

- *Faces*: *Faces* sont les interfaces d'un routeur NDN qui lui permettent de communiquer avec d'autres nœuds NDN. Chaque *Face* possède un identifiant, qui est utilisé par la FIB et la PIT pour transmettre les paquets *Interest* et *Data*, respectivement.

Résolution des noms et routage des données

Chaque routeur NDN intègre des composants pour gérer simultanément la résolution des noms et le routage des données.

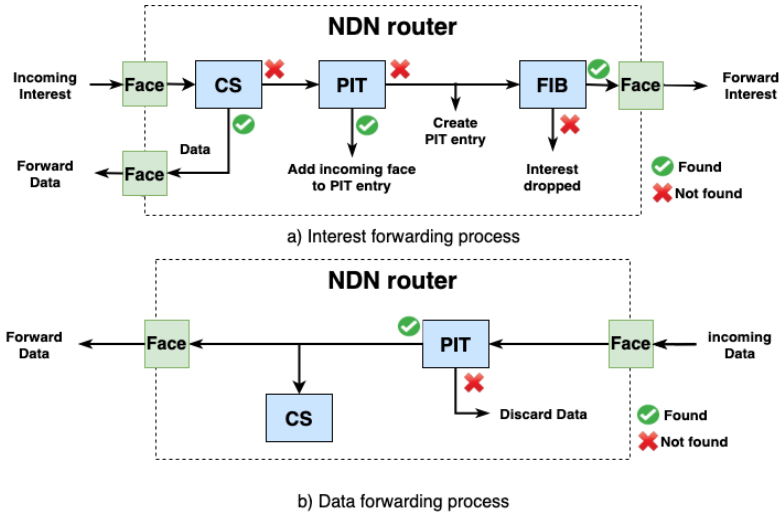


FIGURE A.2: Opérations d'un routeur NDN pour (A) un *Interest* entrant et (B) un *Data* entrant

La figure A.2 illustre le fonctionnement d'un routeur NDN pour le traitement d'un paquet *Interest* et d'un paquet *Data* entrants. Lorsqu'un paquet *Interest* arrive, il est traité par les composants NDN dans l'ordre suivant: le CS, la PIT et la FIB. Premièrement, si une copie en cache correspond au préfixe du paquet, ce qui signifie que les données correspondant au paquet *Interest* sont présentes dans le CS, le routeur NDN transmet immédiatement la copie en cache à la face entrante du paquet *Interest*. Cependant, lorsque le CS ne détient aucune copie, le paquet *Interest* est traité par la PIT. Cette dernière vérifie si une entrée correspond au contenu demandé. D'une part, si une entrée correspond, l'interface du paquet *Interest* est ajoutée à la PIT, et le paquet est supprimé car cela signifie qu'une requête identique a déjà été reçue et transmise, et est éventuellement en attente de son paquet *Data* correspondant. En revanche, si aucune entrée ne correspond au paquet *Interest*, cette dernière est ajoutée à la PIT et transmise à la FIB. Enfin, une fois qu'un *Interest* arrive dans la FIB, si aucune entrée correspondante n'est trouvée, le paquet *Interest* est supprimé. Sinon, l'entrée correspondante guide le routeur NDN pour transmettre le paquet *Interest* à la face correspondante. En suivant la même procédure pour chaque nœud recevant un paquet *Interest*, ce dernier atteint progressivement son fournisseur de données.

En ce qui concerne l'acheminement des données, lorsqu'un paquet *Data* arrive, le routeur NDN vérifie s'il existe une entrée PIT qui correspond au préfixe indiqué. Comme chaque entrée dans la PIT a une durée de vie, elle peut expirer

avant l'arrivée du paquet *Data* qui lui correspond et être supprimée de la PIT, ce qui rend le paquet *Data* entrant non sollicité. Dans ce cas, le paquet est rejeté. D'un autre côté, lorsque les données entrantes correspondent à une entrée dans la PIT, les faces correspondantes, qui ont été référencées dans l'entrée PIT à l'arrivée des paquets *Interest* correspondant, sont utilisées pour transmettre les données aux nœuds en aval. De plus, une copie des données est mise en cache dans le CS pour satisfaire d'autres demandes pour le même préfixe. Encore une fois, en suivant la même procédure pour chaque nœud recevant un paquet *Data*, ce dernier arrive chez son consommateur.

A.2.3 Content Poisoning Attack

La questions de la sécurité d'une architecture innovante telle que NDN est essentielle car son modèle de communication et les composants de son architecture nouvellement introduits exposent le réseau à de nouvelles problèmatiques de sécurité. Parmi ces attaques, l'empoisonnement de contenus (*Content Poisoning Attack (CPA)*) [43] a été identifié comme une menace majeure par la communauté NDN. Dans *CPA*, un attaquant injecte des paquets *Data* corrompus dans le cache des routeurs, afin de résoudre des paquets *Interests* légitimes. CPA exploite la propriété de cache dans le réseau pour diffuser des paquets *Data* corrompus au plus grand nombre d'utilisateurs possible. Afin d'augmenter l'impact de l'attaque, l'attaquant est susceptible de forger des paquets *Data* corrompus avec des noms de contenu populaires.

A.3 Classificateurs de réseaux bayésiens - Etat de l'art

Dans cette section, nous effectuons un état de l'art des classificateurs de réseaux bayésiens (Bayesian Network Classifier - BNC).

A.3.1 Terminologie

Dans [83], Jensen et al. définissent les réseaux bayésiens (Bayesian Network - BN) par (1) un ensemble de variables et un ensemble d'arcs orientés entre les variables ; (2) chaque variable a un ensemble fini d'états mutuellement exclusifs ; (3) les variables et les arcs orientés forment un graphe acyclique orienté ; et (4) à chaque variable A avec les parents $B1, B2$ à Bn est attachée la distribution des probabilités conditionnelles $P(A|B1, B2 \ldots Bn)$. On appelle X_i un parent de X_j (c'est-à-dire $X_i \in Pa(X_j)$) et X_j un enfant de X_i. La relation entre les variables est définie par la distribution de probabilité conditionnelle $\mathbb{P}[X_j|X_i]$ et la distribution précédente du parent X_j. Un *Bayesian Network Classifier (BNC)* est un BN utilisé pour classer l'un de ses nœuds dans un ensemble fini de valeurs. En d'autres termes, BNC est un BN

avec un ensemble de variables aléatoires discrètes $X = (X_1, \ldots, X_n)$ et un ensemble de données observées $E = e$ représentant une *observation*, une variable requêtée X_q, où il faut calculer la probabilité conditionnelle de $P(X_q|E = e)$. Cela signifie que la probabilité conditionnelle de $P(X_q|E = e)$ est la somme de toutes les combinaisons possibles de valeurs des autres variables $X_i \in X - X_q$ de la probabilité conjointe de toutes les valeurs X, sachant $E = e$.

Variable

Une variable aléatoire, appelée X, est un ensemble de valeurs possibles d'un événement aléatoire. Une variable aléatoire peut prendre n'importe quelle valeur dans un ensemble donné.

Nœud - Edge

Dans un BN, chaque nœud correspond à une variable aléatoire unique et chaque arc correspond à une dépendance conditionnelle. Plusieurs événements (c'est-à-dire des nœuds) sont observables, ils sont l'entrée pour effectuer l'inférence et s'appellent les nœuds d'entrée. Tandis que la sortie de l'inférence est la probabilité de différents scénarios d'un événement que nous devons prédire ; les nœuds correspondant s'appellent les nœuds de sortie.

Observation

Une *observation* $E = e$ est un sous-ensemble de variables aléatoires $E = (X_{e_1}, \ldots, X_{e_m})$, où m est le nombre de variables dans le *observation*, représentant l'événement observé, et $e = (x_{e_1}, \ldots, x_{e_m})$ une instanciation de ces variables qui représentent l'occurrence de ces données observées.

Distribution de probabilités conditionnelles

En connaissant la valeur des variables observées, nous devons déduire la probabilité d'autres variables aléatoires dans le réseau. Le coeur de cette déduction réside dans la distribution de probabilité conditionnelle (Conditional probability distribution - CPD). La CPD de Y en sachant X est la distribution de probabilité de Y lorsque la valeur de X est connue, notée $P(Y|X)$.

Facteurs

Un $facteur$ est une fonction qui quantifie l'affinité entre deux variables aléatoires. Soit $(X_1 \ldots X_n)$ un ensemble de variables aléatoires, un $facteur$ ϕ est défini comme une fonction de $Val(X_1, \ldots, X_n)$ dans $\mathbb{R}$. L'ensemble de variables $(X_1, \ldots, X_n)$ est appelé le *cadre* du $facteur$ et est noté $Scope[\phi]$ [82]. Une entrée est un ensemble de valeurs pour chaque variable et la valeur correspondante de ϕ, représentant l'affinité

entre ces valeurs. Plus la valeur ϕ est élevée, plus ces valeurs de variables sont compatibles.

A.3.2 Inférence bayésienne

Dans cette section, nous présentons différents algorithmes qui peuvent réaliser l'inférence bayésienne [82]. L'inférence désigne un algorithme qui consiste à calculer, pour chaque valeur $x_q \in Val(X_q)$, la probabilité de distribution conjointe $P(X_1, ..., X_n)$ puis pour faire la somme des réalisations qui sont cohérentes avec $X_q = x_q$:

$$P(X_q) = \frac{1}{Z} \sum_{X_n} \phi_q \cdot (\ldots (\sum_{X_2} \phi_3 \cdot (\sum_{X_1} \phi_2 \cdot \phi_1))) \tag{A.1}$$

dont

$$Z = \sum_{X_1,\ldots,X_n} \prod_{i=1}^{n} \phi_i(X_i, Pa(X_i)) \tag{A.2}$$

Φ est désigné comme l'ensemble des $facteurs$ dans un BN. Comme nous pouvons le voir, le coeur de l'algorithme d'inférence est de calculer l'expression suivante $\sum \prod_{\phi \in \Phi} \phi$. Le calcul de cette expression nécessite des ressources de calcul importantes. Alors, pour le calculer efficacement, nous devons exécuter le produit dans un sous-ensemble de $facteurs$.

Il existe différents algorithmes capables d'effectuer l'inférence d'un BN ; ces algorithmes sont classés en deux groupes : l'inférence exacte et l'inférence approximative. Dans l'inférence exacte, on calcule la distribution de probabilité conditionnelle sur les variables d'intérêt et on récupère la requête de probabilité conditionnelle requise. Cependant, l'inférence exacte dans les réseaux bayésiens est un problème NP-difficile [99]. Alors, dans certains cas, pour réduire la complexité, on peut utiliser des techniques d'approximation basées sur un échantillonnage statistique, ce qu'on s'appelle inférence approximative [100]. En d'autres termes, le premier donne un résultat exact, mais il est extrêmement coûteux en terme de consomme de ressources. Le second, quant à lui, nécessite moins de ressources, mais le résultat n'est qu'une approximation de la solution exacte. Dans le cadre de cette thèse nous nous sommes concentrés seulement sur l'inférence exacte.

Algorithme d'élimination de variables

L'algorithme d'élimination de variables (Variable Elimination - VE), illustré dans le pseudo-code 2, est la solution fondamentale pour effectuer une inférence bayésienne. L'algorithme calcule certaines sous-expressions et stocke le résultat des calculs intermédiaires pour éviter de générer un nombre exponentiellement élevé d'étapes

de calcul. Le problème central de VE est de calculer le produit-somme, qui est extrêmement coûteux en terme de consommation de ressources. De plus, en pratique, en raison de la structure du réseau bayésien, on peut stocker des calculs qui sont par ailleurs calculés plusieurs fois et ceci exponentiellement.

Plus précisément, l'entrée de l'algorithme VE se compose de deux parties : les $facteurs$ et les *observations*. Les $facteurs$ sont établis par les distributions de probabilités conditionnelles (CPD) et ne changent pas lorsque l'algorithme effectue l'inférence à différents moments, tandis qu'une *observation* $E = e$ est récupérée après chaque itération de l'exécution. La sortie est la probabilité conditionnelle $P(X_q|E = e)$.

Étant donné les données observées (*observation*) $E = e$, nous effectuons une *réduction de facteurs* pour réduire la complexité de chaque $facteur$ dans l'ensemble des $facteurs$ Φ en supprimant les distributions conjointes qui ne correspondent pas aux *observations* (lignes 1-3). Plusieurs variables restent dans la probabilité de requête après la *réduction de facteur* car on ne peut observer qu'un sous-ensemble de variables $E \subset X$. Ces variables seront éliminées par la procédure de synthèse. À cet effet, l'étape suivante consiste à éliminer ces variables selon un ordre d'élimination (ligne 4). Pour chaque variable de l'ordre d'élimination, on multiplie d'abord tous les $facteurs$ qui incluent cette variable, générant un produit de $facteur$ (lignes 6 à 10). Ensuite, on additionne la valeur de la variable et on l'élimine de ce $facteur$ combiné, générant un nouveau $facteur$ qui est entré dans l'ensemble des $facteurs$ à traiter (ligne 11). Par la suite, une fois que toutes les variables ont été éliminées, la seule variable qui reste est la variable de requête X_q. A ce moment, on effectue à nouveau un produit de $facteurs$ pour obtenir le $facteur$ final sur la distribution de X_q (lignes 13-15). Enfin, on normalise le $facteur$ en divisant chaque valeur par leur somme (lignes 16-17).

Algorithm 2 Algorithme Variable Elimination - VE

Input: *initial facteurs* (Φ) and *observation* (E=e)

Output: Conditional probability $P(X_q|E = e)$

foreach $\phi_i \in \Phi$ **do**
  $\phi_i \leftarrow \phi_i(E = e)$ // Factor reduction
end
Select Elimination Order (σ);
foreach $x_i \in \sigma$ **do**
  foreach $\phi_j \in \Phi$ **do**
    if $x_i \in Scope[\phi_j]$ **then**
      $\psi_i \leftarrow \psi_i * \phi_j$ // Factor product
    end
  end
  $\phi_i \leftarrow \sum_{X_i} \psi_i$ // Factor marginalization
end
foreach $\phi \in \Phi$ **do**
  $\varphi \leftarrow \varphi * \phi$ // Factor product
end
$Z \leftarrow \sum_{X_1 \ldots X_n} \varphi$
$P \leftarrow \varphi / Z$ // Factor normalization

A.4 Un plan de surveillance pour NDN

Dans cette section, nous présentons le premier élément de notre travail : un plan de surveillance pour NDN. Les travaux présentés dans cette section ont été menés en collaboration avec Ngoc-Tan Nguyen et les contributions décrites dans la section ont été présentées sous forme de d'article dans [121] et un chapitre dans la thèse de Ngoc-Tan Nguyen [122] .

A.4.1 Métriques surveillées

Nous définissons un ensemble complet de métriques pour révéler tout anomalie potentielle dans le réseau. Pour établir cet ensemble de métriques, nous avons considéré tous les composants pertinents à l'intérieur d'un nœud NDN, y compris (1) les *Faces*; (2) le *Content Store* (CS); (3) la *Pending Interest Table* (PIT) et (4) la *Forwarding Information Base* (FIB). Nous avons délibérément décidé de ne pas inclure la FIB dans notre champ de métriques considérées. En effet, contrairement aux autres éléments inclus dans notre sélection, la FIB appartient au plan de contrôle et ses modifications ne sont attribuées qu'aux configurations de routage statique ou aux mises à jour du protocole de routage et nous avons restreint notre champ d'étude à la seule

surveillance du plan de données.

Le nombre de paquets entrants et sortants dans une période d'échantillonnage, y compris *In Interest*, *In Data*, *In NACK*, *Out Interest*, *Out Data*, *Out NACK*, sont des métriques naturelles pour décrire l'activité des composants. Avec le nombre de paquets entrants et sortants, le nombre de paquets abandonnés (c'est-à-dire *Drop Interest*, *Drop Data*, *Drop NACK*) est également pris en compte puisque le routeur peut supprimer des paquets selon sa stratégie. À l'arrivée d'un paquet *Interest*, le contenu demandé est vérifié dans le cache local du routeur NDN, appelé CS. Cette opération se termine soit par un échec de cache, soit par un hit de cache. Sur la base de la capacité limitée du cache et de la politique de remplacement du cache, le CS décide d'insérer un nouveau *Data* valide dans son stockage ou de supprimer les données mises en cache lorsqu'il est plein.

la PIT représente une table dans laquelle un routeur NDN suit les *Interest* valides qu'il a transmis et qui n'étaient pas encore satisfaits par le réseau. Ces entrées sont également utilisées pour inverser le chemin des paquets *Data*. De la même manière que CS, le nombre d'entrées créées (*PIT Create*), supprimées (*PIT Delete*), mises à jour (*PIT Update*) ainsi que le nombre actuel d'entrées (*PIT Number*) sont susceptibles d'être suivis régulièrement depuis le PIT. En outre, un paquet *Interest* a une durée de vie qui correspond approximativement à son temps de stockage dans la PIT avant l'arrivée d'un paquet *Data* ou avant d'être supprimé par le routeur. Une expiration de la durée de vie d'un paquet *Interest* décrit une situation anormale qui doit également être surveillée à la fois pour des raisons de performances et de sécurité. Nous agrégeons le nombre de *PIT Unsatisfied* dans n'importe quel nœud.

Nous avons créé également la métrique *PIT Exist Time* qui contient la moyenne de chaque valeur considérée dans la période d'échantillonnage pour présenter l'état des entrées dans la PIT sur le long terme.

A.4.2 Sonde de surveillance NDN

L'extraction des métriques précédemment décrites est effectuée par l'outil de surveillance Montimage (MMT) [1], qui est dédié à la surveillance du trafic réseau, depuis des fichiers journaux et des traces d'application. Cet outil permet de dériver divers événements et métriques du réseau, des services et du système en fonction des besoins des utilisateurs final qui administre les composants.

A.4.3 Micro détecteur

Au delà de cette instrumentation, l'étape suivante vers la conception d'un plan de surveillance, consiste à combiner les mesures fournies par chaque métrique pour

[1] http://www.montimage.com/products.html

obtenir la vue la plus la plus pertinente sur l'événement que l'on souhaite détecter et à concevoir une méthode qui décide quand une alarme d'événement anormal doit être déclenchée. Les deux principales difficultés dans la conception d'un tel détecteur sont de pouvoir s'adapter à l'aspect dynamique naturel du trafic réseau et de pouvoir modéliser à la fois le comportement normal et anormal de toutes les métriques.

L'idée générale de la méthode que nous avons utilisée pour concevoir nos micro-détecteurs est basée sur des tests de Constant False Alarm Rate (CFAR) qui visent à (1) répondre à une probabilité déterminée a priori de fausse alarme et (2) maximiser la probabilité de détection d'événements anormaux sous sous cette contrainte de fausse alarme. Le contrôle de la probabilité de fausse alarme est d'un intérêt crucial d'un point de vue opérationnel pour éviter de nombreuses fausses alarmes et donc pour avoir une détection fiable des événements anormaux. Il convient de noter qu'en pratique, il n'est guère possible de maximiser la probabilité de détection d'événements anormaux lorsque de nombreux événements différents sont pris en compte. Par conséquent, la plupart de nos micro-détecteurs sont des tests bilatéraux basés sur un modèle statistique de trafic légitime. Sur un ensemble de mesures, une anomalie devrait déplacer un indicateur résultant dans une direction spécifique. Par exemple, le taux de paquets *Interest* non satisfaits ne peut être diminué qu'en cas de dysfonctionnement. Dans ce cas un détecteur unilatéral plus puissant peut être conçu. Des exemples de micro-détecteurs d'attaque NDN qui sont construits sur de tels modèles statistiques et qui peuvent s'adapter à l'aspect dynamique du trafic peuvent être trouvés dans [154].

A.4.4 Un classificateur de réseau bayésien

Les micro-détecteurs basés sur une seule métrique ne peuvent décrire qu'un aspect spécifique de l'état d'un nœud NDN. Cependant, différents types d'attaques ont des effets différents sur les métriques. De plus, le changement d'une métrique ne peut pas, à lui seul, conclure sur la situation d'un nœud NDN. Un exemple simple illustrant cette limitation est le cas où une augmentation standard du trafic peut affecter la métrique *Interest* et peut être confondue avec une attaque. C'est pourquoi les alertes provenant d'un seul micro-détecteur ne peuvent pas être interprétées comme une attaque sur un nœud NDN et une corrélation des alertes micro-détecteurs est nécessaire. Afin de démontrer les relations causales entre les micro-détecteurs, une structure de réseau bayésien est proposée et illustrée dans Figure A.3. Chaque nœud correspond au micro détecteur d'une métrique. Il y a deux raisons principales qui justifie le choix d'un réseau bayesien. Premièrement, un BN peut corréler un ensemble de métriques pour détecter si un événement d'anomalie se produit. Deuxièmement, un BN est une méthode probabiliste bayésienne et les métriques mesurables

des réseaux informatiques ne sont pas entièrement prévisibles. Ainsi, le BN peut efficacement gérer la nature aléatoire sous-jacente des métriques observées.

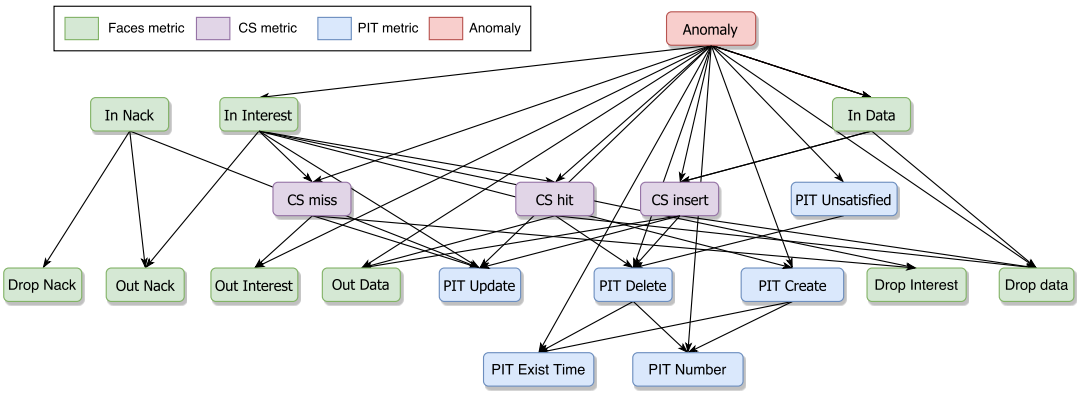


FIGURE A.3: Le réseau bayésien.

Dans notre BN, la détection des anomalies pouvant survenir dans le réseau NDN est représentée par le nœud *Anomaly*. Les arcs orientés dans le BN reflètent la relation causale entre les paires de métriques ou entre le nœud *Anomaly* et une métrique. L'arc part du nœud qui affecte celui qui est impacté. Étant donné que la relation de causalité dans chaque BN est construite sur la base d'une connaissance experte de la relation entre les nœuds dans le BN, nous choisissons délibérément les pipelines de transfert NFD pour déduire la relation de causalité, car tous les changements de métriques se produisent au sein ces pipelines. Un pipeline de transfert est une série d'étapes qui opèrent sur un paquet ou une entrée PIT provoquée par un événement spécifique [124].

A.4.5 Evaluation

Nous évaluons la pertinence de notre approche dans le contexte particulier de l'attaque CPA. CPA est une attaque typique dans NDN et a un impact sur plusieurs métriques, tandis que les changements normaux dans le comportement du réseau observé n'affectent souvent qu'une seule métrique ou un sous-ensemble des métriques. Le CPA est donc un bon candidat pour la validation.

Les résultats expérimentaux ont été obtenus dans un environnement virtuel qui reproduit la topologie présentée dans Figure A.6. La topologie contient trois routeurs : un routeur périphérique côté client R1, un routeur principal R2 et un routeur R3 qui représente le routeur périphérique et le système de mise en cache côté fournisseur légitime. Les clients légitimes et les attaquants se connectent à R1. Dans chaque routeur, une sonde MMT est déployée pour collecter des données pour les 18 métriques sélectionnées. La sonde MMT est utilisée pour collecter des données à partir du journal NFD. Les données sont traitées par un micro-détecteur, qui récupère l'état du nœud dans la base de données locales. Enfin, les résultats des

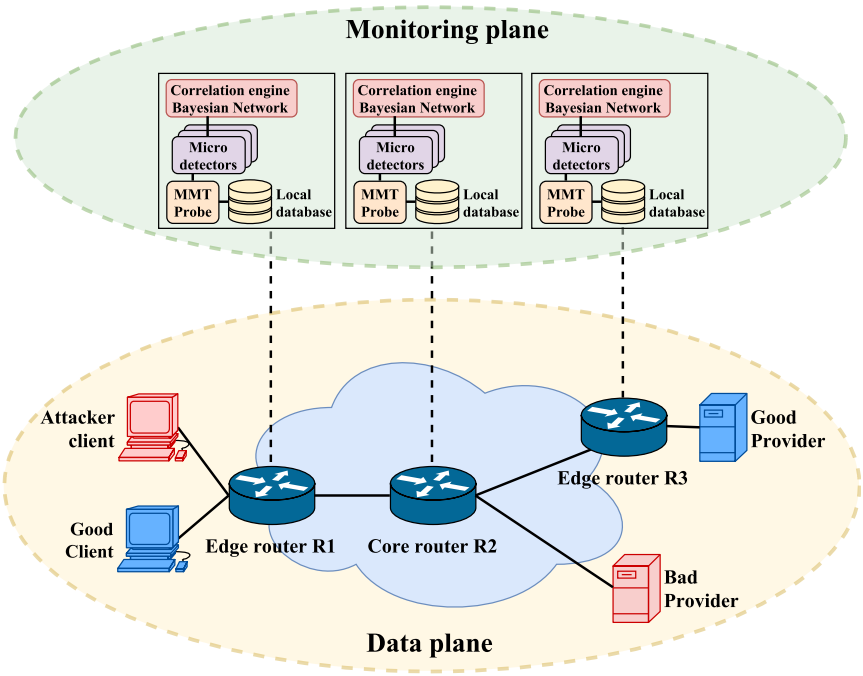
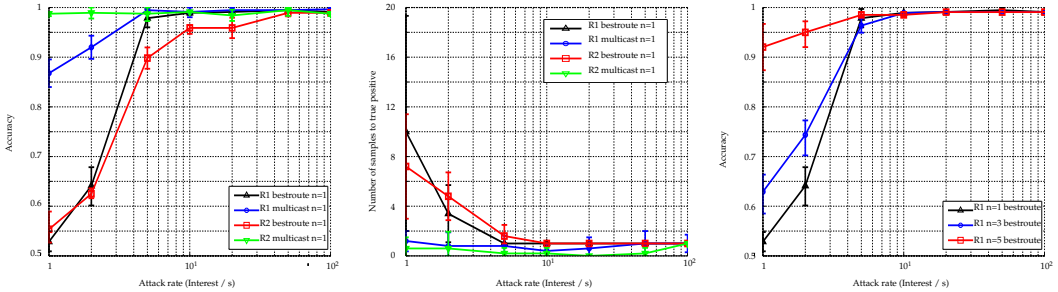


FIGURE A.4: Topologie de cas d'utilisation pour Content Poisoning Attack.



(A) Précision (B) Délai de détection de CPA (C) Fenêtre de détection différente

FIGURE A.5: Performances du BNC proposé

micro-détecteurs sont agrégés par le *Bayesian Correlation Engine Network* pour déterminer l'état du nœud.

Figure A.5a et A.5b représentent respectivement la précision et le retard du BNC lorsque le taux d'attaque change. En ce qui concerne l'impact du taux d'attaque, les chiffres indiquent que lorsque le taux d'attaque est inférieur au taux de *Interest* dans le trafic normal, le BNC a une faible précision, un retard élevé et ces résultats ont une large diffusion statistique.

En ce qui concerne l'impact du scénario d'attaque, on peut noter à partir de Figure A.5a et A.5b que la précision et le retard du BNC contre *CPA multicast* sont meilleurs que dans *CPA bestroute*. La raison en est que, dans *CPA multicast*, les *Interests* sont plus susceptibles d'être transmis au mauvais fournisseur sans trop d'effort de la part de l'attaquant, de sorte que le comportement d'attaque quel que soit le taux est visible. Concernant l'impact de l'emplacement, dans *CPA bestroute*, le BNC

au niveau de R1 est plus précis que dans le routeur R2 car il reçoit d'abord les paquets *Interest* des clients et des attaquants. En conséquence, ses métriques seront plus affectées que celles de R2. Tandis que dans *CPA multicast*, étant donné que R2 est plus susceptible d'être empoisonné en raison de l'avantage du délai du mauvais fournisseur, ses mesures sont affectées plus clair que celles de R1. Par conséquent, le BNC dans R2 atteint une meilleure précision que dans R1. Finalement, il est important de souligner que la précision du BNC peut être améliorée en augmentant la fenêtre de détection n du micro détecteur. Comme indiqué dans Figure A.5a, pour le pire cas de *CPA bestroute* avec $1 Interest/s$, la précision est passée de 53 % à 93 % en augmentant n de 1 à 5. Néanmoins, il convient de noter que l'augmentation de la fenêtre de détection augmentera le délai de détection des micro-détecteurs. Ce compromis doit être soigneusement pris en compte lors du déploiement.

A.5 Vers une orchestration orientée contenu pour la sécurité

Dans cette section, nous proposons une orchestration orientée contenu pour la sécurité de NDN basée sur des architectures standardisées. L'architecture globale que nous avons mise en œuvre suit strictement la spécification d'architecture de référence ETSI [33] et considère Docker, une solution efficace et bien adoptée, comme technologie de base pour *Network Functions Virtualization Infrastructure (NFVI)*. Le *Virtual Infrastructure Manager (VIM)* chargé de contrôler et de gérer les ressources de calcul, de stockage et de réseau *NFVI*, n'a pas besoin d'être étendu pour prendre en charge le trafic NDN. Par conséquent, nous avons sélectionné Docker Swarm comme technologie prête à l'emploi.

A.5.1 Profil et politiques NDN TOSCA

Notre profil TOSCA étend le profil simple OASIS pour NFV afin de prendre en compte les fonctionnalités NDN. Fondamentalement, une spécification TOSCA fournit les nœuds de base et les politiques suivantes pour construire une topologie de service : *Virtual Deployment Unit (VDU), VNF, Virtual Link (VL), Connection Point (CP), Forwarding Path* et politiques. Dans notre cas, nous réutilisons les trois nœuds standard suivants sans aucune modification: *VDU, VL* et *CP* pour modéliser un réseau NDN virtualisé, car ils ne concernent que la couche d'infrastructure agnostique à NDN. Au contraire, pour modéliser les éléments de transmission d'un réseau NDN virtualisé, les nœuds TOSCA pour *VNF*, *Forwarding Path* et les politiques ont été étendus.

Notre nœud *VNF* comprend des paramètres de configuration qui représentent l'ensemble des préfixes NDN à annoncer ainsi que l'état d'un module de vérification de signature dans un routeur NDN. Nous avons également inclus des informations

supplémentaires dans le nœud *VNF* pour prendre en compte les propriétés spécifiques de composants NDN particuliers. Deuxièmement, la spécification *Forwarding Path* a également été étendue pour capturer la liste des VNF qu'un ensemble particulier de paquets NDN suivra.

Notre extension NDN TOSCA active les politiques de spécification modélisées avec des règles ECA (Event-Condition-Action) qui s'appliquent dynamiquement pendant l'exécution du service. Le profil NDN TOSCA comprend deux types de politiques qui traitent des opérations de reconfiguration dynamique et du passage à l'échelle des composants (*scale-out*).

A.5.2 Composants compatibles NDN

La méthodologie que nous avons suivie a consisté à étendre ou à reconcevoir uniquement les composants MANO nécessitant une prise de conscience NDN, sans les détourner de leur objectif initial. Il s'agit du *VNF Manager (VNFM)* et du *NFV Orchestrator (NFVO)*. Dans notre architecture, les VNF sont des applications Docker, qui incluent un routeur NDN et un pare-feu NDN. Dans NDN, les principales fonctionnalités du *NFVO* doivent être repensées pour prendre en compte les caractéristiques de ce nouveau paradigme. Pour les intégrer, nous avons conçu et implémenté un *NFVO* dédié qui lit les modèles TOSCA et crée un graphe en mémoire des nœuds TOSCA et de leurs relations et en particulier la configuration NDN dans un profil NDN TOSCA. Outre la configuration initiale définie dans la spécification de déploiement d'un profil NDN TOSCA, le *NFVO* prend également en compte les actions à effectuer à partir des politiques NDN TOSCA. Dans le contexte de NDN, le *VNFM* est étendu pour transférer toutes les configurations spécifiques NDN du *NFVO* vers les *NDN VNFs*. Il est également étendu pour recevoir des notifications des différents *Element Managers (EM)*, qui sont les composants internes responsables de toutes les opérations de gestion dans une *NDN VNF*.

A.5.3 Orchestration orientée contenu

Dans la phase de déploiement, premièrement, le *VIM* déploie un réseau de gestion pour assurer la communication entre les VNF et le *VNFM*, suivi du déploiement du conteneur de *VNFM* se connectant à ce réseau. Ensuite, le *VNFM* met à jour les routes pour les préfixes dans les *FIB* grâce aux *EM*. Un problème qui peut se produire à ce stade est que la connexion entre les VNF peut ne pas être prête lorsque le routeur termine de recevoir sa configuration NDN du *VNFM*. Pour résoudre ce problème, périodiquement, chaque *EM* vérifie la *FIB* de son routeur NDN associé pour s'assurer que le processus *NFD* prend déjà en compte la configuration reçue du *VNFM* et il procède à une reconfiguration si besoin. Déclenché par un événement défini dans les politiques NDN TOSCA, le *NFVO* peut également reconfigurer

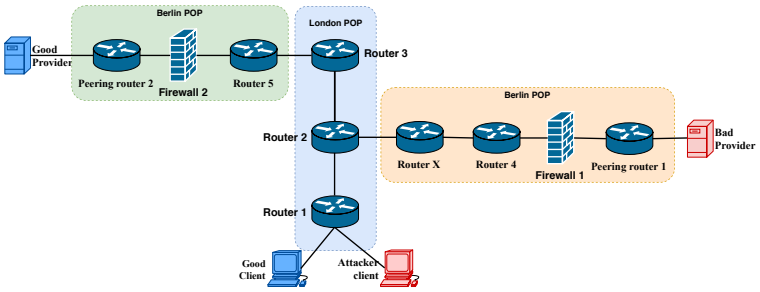
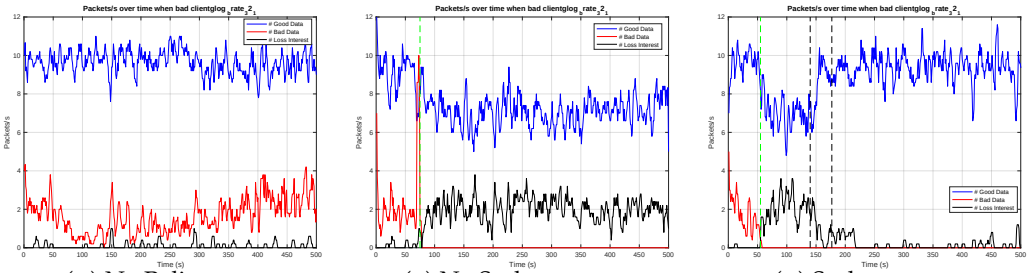


FIGURE A.6: Experiment topology



(A) No Policy (B) No Scale-out (C) Scale-out

FIGURE A.7: Snapshot of Good/Bad/Lost data in 3 scenarios

dynamiquement toute fonction NDN virtualisée avec un processus similaire au déploiement.

Le passage à l'échelle d'un service (*scale-out*) est une caractéristique essentielle d'une architecture MANO pour garantir les performances et le débit du réseau. Pour faciliter la compréhension, nous l'illustrons dans le contexte d'un routeur *SV-Router* qui effectue une vérification de signature et dont la charge devient par conséquent élevée. *SV-Router* envoie périodiquement le nombre d'entités de sa *PIT* au *VNFM* qui à son tour le transmet au *NFVO*. A chaque fois qu'un seuil donné est franchi, le *NFVO* ordonne au *VIM* et au *NFVI* d'augmenter les *containers* de VNF à un nombre spécifié de répliques qui est défini dans le profil TOSCA. Enfin, la stratégie de transmission dans ce nœud est modifiée en mode *round-robin*, ce qui permet d'équilibrer la charge dans les différentes répliques du *SV-Router*. Au fur et à mesure que le processus de montée en puissance se déroule progressivement, le réseau continue de fournir le service sans interruption.

A.5.4 Évaluation

Use case

Dans la topologie que nous considérons, illustrée dans la figure A.6, un fournisseur légitime NDN est connecté au *Peering router 2* tandis qu'un fournisseur malveillant est connecté au *Peering router 1*. Le *router 2* connecte à la fois l'utilisateur légitime et l'attaquant dont le but consiste à corrompre les données hébergées dans tous les caches réseau pour empêcher l'utilisateur légitime d'accéder aux *Data* souhaitées. C'est le scénario d'une attaque par empoisonnemment de cache (CPA) qui permet

d'évaluer la partie dynamique de notre framework d'orchestration.

Scénarios d'orchestration

La détection de cette attaque est réalisée grâce à un détecteur dédié présenté dans [155] et du point de vue de l'orchestration, nous considérons trois scénarios d'atténuation : *No policy*, *No scale-out* et *Scale-out*. Dans le cas de *No policy*, l'ensemble du réseau est déployé sans aucune politique pour atténuer les menaces de sécurité. Ce scénario est considéré comme une référence pour évaluer la pertinence des suivants. Dans le deuxième scénario (*No scale-out*), le *NFVO* active une politique qui permet la configuration dynamique du pare-feu et l'activation dynamique de la vérification de signature dans les routeurs NDN. Enfin, dans le scénario *Scale-out*, le *NFVO* applique les politiques d'atténuation présentées ci-dessus et il applique également une opération de *scale-out* sur les routeurs NDN qui souffrent d'une charge de calcul trop élevée.

Les paramètres constants qui ont été utilisés pour exécuter ces trois scénarios sont réutilisés à partir d'expériences faites dans [156] et les scripts suivants illustrent les politiques qui ont été utilisées dans le cas *Scale-out*. La politique de vérification de signature est déclenchée par le *Router 2* pour activer la vérification de signature dans les cibles *Router 4* et *Router 5* qui sont les meilleurs candidats pour effectuer cette vérification qui peut empêcher les *Data* malveillants d'être poussés par le mauvais fournisseur. Ensuite, la politique de pare-feu est déclenchée par *Router 4* et *Router 5* et elle permet à la configuration de *Firewall 1* et *Firewall 2* d'être dynamiquement augmentée avec une règle qui consiste à bloquer les préfixes des paquets *Data* malveillants. Enfin, la politique suivante fournit un exemple de politique de performance qui permet de mettre à l'échelle les VNF *Router 4 Router 5* à trois répliques chaque fois qu'un seuil donné est franchi, qui est ici le nombre d'entrées dans la PIT de routeurs NDN.

```
scaling_out_policy:
  type: tosca.policies.nfv.doctor.ndn.scaling
  cibles: [router_4, router_5]
  déclencheurs:
  scale_out:
      nom_mètre: PIT
      type_événement: tosca.policies.nfv.doctor.ndn.n_pit
      état:
      contrainte: pending_interests Greater_than 5
      seuil: 5
      opérateur_comparaison: gt
      période: 10
      action:
      type_action: scale_out
      numéro 3
```

Résultats numériques

La figure A.7 présente des séries chronologiques de *Good Data* (lignes bleues), *Bad Data* (lignes rouges) et *Lost Data* (lignes noires), exprimées en termes de nombre de paquets par seconde, reçus par l'utilisateur légitime dans le cadre des trois scénarios décrits ci-dessus.

Dans le scénario *No policy*, la fréquence de *Good Data* est d'environ 10 paquets/s. Cependant, il existe également une quantité substantielle de *Bad Data*, environ 3 paquets/s. Ce scénario montre que sans atténuation, l'attaque peut corrompre jusqu'à un tiers des paquets *Data* fournis à un utilisateur légitime.

Dans le scénario *No scale-out*, la ligne verte indique l'heure à laquelle le réseau détecte l'attaque et déclenche le pare-feu et les politiques de vérification de signature pour atténuer l'attaque. Nous pouvons observer qu'après ce moment, le nombre de *Bad Data* tombe brusquement à zéro. Cependant, le nombre de *Good Data* diminue également légèrement et le nombre de *Lost Data* augmente considérablement. La raison de ce phénomène réside dans la surcharge dont souffrent les routeurs effectuant la vérification de signature, qui l'empêchent de réaliser de manière fiable ses opérations de transmission.

Dans le dernier scénario, *Scale-out*, les deux lignes sombres indiquent l'instant auquel les opérations de configuration de scale-out commencent et se terminent. Dans ce scénario, on remarque que l'efficacité d'atténuation est identique au scénario *No scale-out*, avec un excellent filtrage des données malveillantes. De plus, lorsque le *scale-out* est appliqué, le réseau revient à un état normal : le taux de *Good Data* revient à 10 paquets/s, et *Lost Data* tombe à une valeur négligeable. On peut également observer que pendant la période de montée en puissance, il n'y a pas de temps d'arrêt du réseau.

Au-delà de ces illustrations d'atténuation, dans toutes nos expériences, le pourcentage de *Bad Data* dans les scénarios *No Scale-out* et *Scale-out* est plus petit que dans *No Policy* qui démontre l'efficacité de la vérification de signature et des politiques de reconfiguration du pare-feu. Le pourcentage de *Bad Data* qui reste sur ces deux scénarios est le pourcentage de *Bad Data* avant la détection d'attaque. Après l'activation de ces politiques, le nombre de *Bad Data* diminue.

En conclusion, les résultats numériques montrent que les politiques visant à remédier à l'attaque et aux incidents de performance sont efficaces.

A.6 Une architecture de détection d'anomalies orientée contenu

Dans cette section, nous abordons le dernier défi concernant le plan de sécurité, qui est la performance des fonctions de sécurité et comment optimiser l'utilisation des ressources informatiques tout en les exécutant. Puisqu'un plan de sécurité surveille en permanence un réseau à grande échelle, un grand nombre d'opérations de sécurité sont répétées. En particulier, il apparaît que chaque nœud effectue les mêmes tâches de sécurité dans le temps, en particulier dans une situation normale où il n'y a pas d'attaques ou de changements importants. En outre, certains des résultats des fonctions de sécurité ont probablement déjà été calculés par les nœuds voisins. À cet égard, nous exploitons la fonctionnalité de mise en cache dans le réseau de *Named Function Networking - NFN*, qui offre un moyen substantiel pour réduire les ressources de calcul des fonctions de sécurité.

A.6.1 Named Function Networking

L'architecture NFN (Named Function Networking), inspirée par le concept des réseaux actifs, a été introduite dans [132, 133] pour étendre CCN/NDN avec des noms des fonctions de calcul sur les contenus. Dans NFN, au lieu de demander des données, un utilisateur envoie un intérêt au réseau avec l'expression d'une fonction pour la récupération de son résultat. Les données renvoyées aux utilisateurs sont plutôt les résultats calculés, qui peuvent également être mis en cache comme tout paquet de données. Les programmes fonctionnels dans NFN sont exprimés en λ-calculus [134], spécifiés dans un espace de noms hiérarchiques et intelligibles, et transportés dans un paquet *Interest*.

Le fonctionnement de NFN s'appuie sur un moteur de résolution d'expressions λ-calculus dans chaque nœud avec éventuellement un serveur de calcul ou une application hébergée. De façon similaire à un fournisseur de contenu NDN, si un nœud NFN héberge un serveur de calcul, sa fonction nommée est publiée comme une expression de λ-calculus dans le réseau, et les intérêts peuvent lui être transmis. Les routeurs utilisent la recherche par corresponsance de nom la plus longue pour transférer les *Interest*. Si une copie du résultat d'une fonction est mise en cache sur un nœud intermédiaire, elle sera renvoyée à l'utilisateur demandeur. Sinon, l'un des nœuds NFN, qui a reçu le *Interest* sur le chemin d'accès à la source de données, tentera de calculer le résultat en fonction des stratégies et des ressources de traitement disponibles.

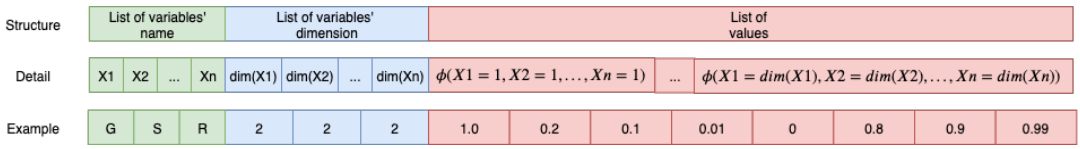


FIGURE A.8: Structure de données d'un *facteur*

A.6.2 Nommage et structure des données

Comme illustré dans la section A.3, la notion de *facteur* est le cœur de l'algorithme d'inférence. Un *facteur* n'est pas seulement l'entrée mais aussi la sortie des fonctions. De plus, le paramètre *observation* peut également être utilisé pour calculer la *Réduction de facteurs*. Les structures de *facteur* et *observation*, leurs schémas de dénomination et le schéma de dénomination des fonctions sont décrits dans la section suivante.

Un paquet *facteur* comprend trois parties: la liste des noms des variables aléatoires, leurs dimensions et la liste de toutes les valeurs de ϕ. Les deux premières parties se réfèrent aux métadonnées du paquet, tandis que la troisième partie correspond aux données réelles. La liste de toutes les valeurs est triée dans l'ordre d'un *facteur*. La figure A.8 illustre cette structure.

Plus précisément, les structures de données proposées incluent deux types de *facteurs* : *facteurs initiaux* et *facteurs temporaires*. Les *facteurs initiaux*, qui sont établis par les CPD, sont nommés comme suit : */data/fac/initial/<name of variables >*. Par exemple, */data/fac/initial/GzSzR*, avec "z" est la caractère séparateur, est le nom du *facteur* $\phi_3(G, S, R)$. Les *facteurs* sont également les sorties des opérations dans l'algorithme d'inférence, puis les entrées pour les opérations suivantes. Par conséquent, *facteurs temporaires* sont utilisés pour répartir le calcul entre les nœuds d'un réseau NFN. Ces *facteurs* sont nommés en utilisant le hachage de leur contenu : */data/fac/temporaire/<(factor)>*. Une alternative pour nommer un *facteur temporaire* consiste à utiliser ses valeurs en tant qu'un nom. Cependant, lorsqu'il est complexe, le nom peut devenir trop long pour être stocké dans un paquet NFN. Un autre choix consiste à générer le nom du *facteur temporaire* de façon aléatoire ou incrémentale. Cependant, si nous nommons les *facteurs* dans chaque nœud NFN de cette façon, lorsqu'un nœud demande à d'autres de calculer une opération mais un autre nœud utilise déjà ce nom pour un autre *facteur*, le résultat sera inexact. Les fonctions de hachage évitent une telle collision car le même hachage signifie que les mêmes données sont stockées dans le cache.

De même, un paquet d'une *observation* comprend trois parties : le nom des variables, leurs dimensions et les valeurs d'une *observation*. La structure de données d'une *observation* est conçue en tenant compte des idées suivantes. Premièrement,

l'*observation* d'une variable $X_e = x_e$ montre que la valeur de la variable X_e est connue (x_e). Alors, les entrées qui englobent les valeurs $X_e \neq x_e$ seront supprimées lors de l'introduction d'une *observation*. Pour atteindre cet objectif, la valeur d'une *observation* pour une variable dans un paquet consiste en une chaîne de 0 et de 1. La valeur 0 signifie que l'entrée correspondante sera supprimée tandis que la valeur 1 signifie que l'entrée restera pendant l'introduction d'une *observation*. Comme illustration de l'*observation* $G = 2$, la valeur du paquet *observation* est 01. Deuxièmement, une *observation* ne se compose pas nécessairement que d'une seule variable, elle peut être plutôt composé de plusieurs variables. Par conséquent, les données d'une *observation* sont la fusion des valeurs de chaque variable dans l'*observation*. Comme illustration de l'*observation* $G = 2, S = 1$, la valeur du paquet *observation* est 01z10. Enfin, une *observation* est conçue pour être calculé avec un $facteur$. Certaines variables appartiennent à E, et d'autres variables appartiennent à un $facteur$ qui doivent tous deux être présents dans la valeur d'une *observation*. Les valeurs de ces variables sont encore inconnues. Cela signifie qu'elles resteront tout en introduisant les *observations*. Elles sont donc marquées par une chaîne de valeurs 1. Par exemple, la valeur de l'*observation* $G = 2$ dans le contexte du $facteur$ $\phi_3(G, S, R)$ est 01z11z11. La figure A.9 illustre la structure des données de l'*observation* $G = 2$ dans le contexte du facteur $\phi_3(G, S, R)$.

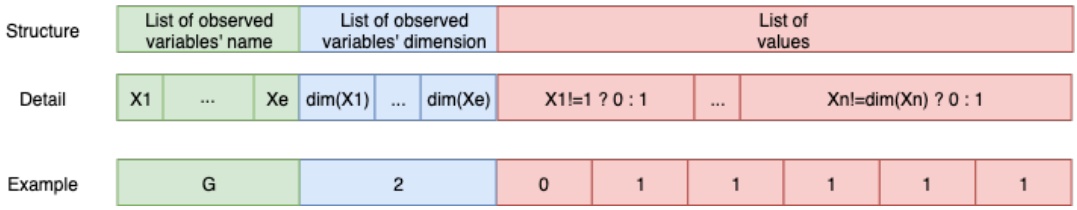


FIGURE A.9: Structure des données d'une *observation*

Enfin, les préfixes */func/<nom des fonctions >/* sont réservés aux fonctions NFN. Ces fonctions sont identifiées par leurs noms et paramètres (où un paramètre peut être une chaîne, un entier ou un nom de données). Par exemple, la fonction *réduction de facteurs* des algorithmes VE est: */func/reduce/(/data/fac/.../data/evi/...)*

A.6.3 Transformation de fonctions en λ-calculus

Nous choissions la réduction de facteur pour expliquer en détail comment nous transformons ces fonctions en λ-calculus, puis comment nous exploitons NFN.

Soit $\phi(X)$ un $facteur$, et une *observation* $E = e$. Nous définissons la réduction du $facteur$ ϕ au contexte $E = e$, noté $\phi[E = e]$. Pour atteindre cet objectif, nous devons examiner comment nous introduisons une *observation*. La fonction *réduction de facteurs* est construite sur l'idée d'effacer les entrées invalides qui entrent en conflit avec l'*observation* introduite. Par exemple, étant donné les observations $X_e = x_e$ et

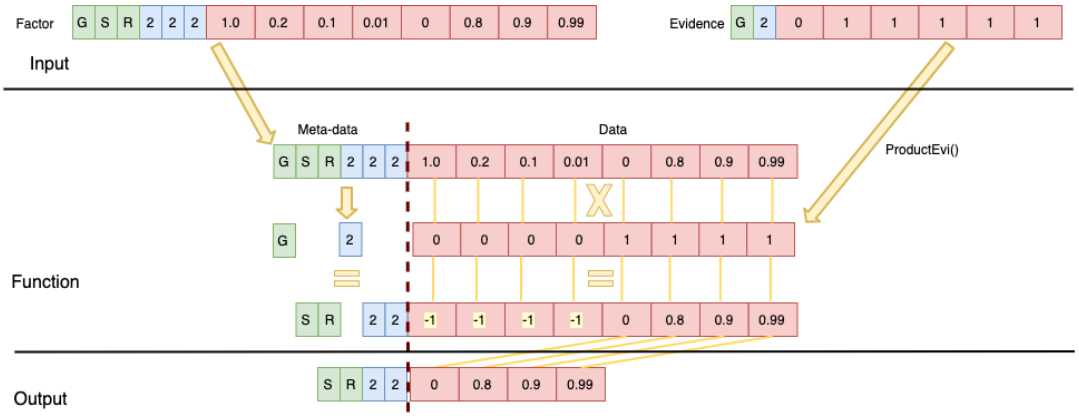


FIGURE A.10: Exemple de fonction *réduction de facteurs*

un $facteur\ \phi(X_1 \ldots, X_n)$, le nouveau $facteur\ \phi(X_1, \ldots X_{e-1}, X_e = x_e, X_{e+1}, \ldots, X_n)$ est construit en supprimant les entrées qui incluent $X_e \neq x_e$, comme nous avons expliqué dans la section A.3.

La valeur d'une *observation* dans notre approche est conçue pour indiquer que l'entrée doit être supprimée dans le $facteur$. Pour identifier l'entrée qui doit être supprimée, nous utilisons la valeur 0, et pour l'entrée qui doit être conservée après l'opération, la valeur 1. Comme représenté sur la figure A.8, la troisième partie du $facteur$ est une sérialisation des valeurs d'un $facteur$ dans le bon ordre. Cependant, la valeur d'un packet *observation* n'est qu'une fusion des différentes *observations*. Alors, il faut construire une fonction qui permet de fusionner les *observations* des variables en une donnée intermédiaire compatible avec la dimension et le bon ordre du $facteur$ à réduire. La fonction *productEviVal* permet de fusionner *observations* à partir de toutes les variables aléatoires dans le $facteur$ et de fournir de nouvelles données compatibles avec le $facteur$. L'idée de l'algorithme est que nous réalisons le produit entre les valeurs dans l'*observation* de deux variables (ligne 2). En particulier, si la valeur dans l'une des variables est déjà marquée 0 pour être supprimée, cela signifie également que l'entrée constituée de cette valeur avec toutes les valeurs des autres variables sera marquée 0 et sera également supprimée.

Function *productEviVal(evi1,evi2)*
| return reduce(lambda x,y:x+y,[[i*j for i in evi1] for j in evi2])

Ensuite, l'algorithme suivant permet l'introduction d'une *observation* dans un $facteur$. Premièrement, la fonction *getVarDim* permet de fusionner les noms de variables et leurs dimensions de $facteur$ et *observation* en une seule liste, qui est la métadonnée du nouveau $facteur$ (ligne 2). Deuxièmement, les valeurs des $facteurs$ et *observation* sont calculées pour établir les données du nouveau $facteur$. Si la valeur d'une *observation* est 0, cela signifie que l'*entre* du $facteur$ sera supprimée (ligne 4). Le résultat sera marqué -1, car la valeur du $facteur$ valide est toujours positive, et

cette entrée sera supprimée plus tard. En revanche, si la valeur d'une *observation* est 1, ce qui signifie que l'entrée est toujours valide, la valeur du nouveau *facteur* est conservée en la multipliant par 1 (ligne 4). Enfin, toutes les valeurs non valides du *facteur*, c'est-à-dire celles qui sont négatives, sont supprimées pour obtenir finalement un *facteur* valide (ligne 3). La figure A.10 illustre ce processus.

Function *reduceFactor(fac,evi)*

return getVarDim(fac,getVarDimEvi(evi)) + SEPARATOR_CHAR serialize(filter(lambda x:x >-1, map(lambda x,y: -1 if y == 0 else x*y, deserialize(fac), reduce(lambda x,y: productEviVal(x,y),getValEvi(evi)))))

A.6.4 Evaluation

Pour évaluer les performances de notre approche, le cas d'utilisation CPA est considéré. La détection de cette attaque est réalisée grâce au détecteur dédié présenté dans A.4. Une topologie contenant trois routeurs NDN avec à la fois un redirecteur NFN et un serveur de calcul NFN est utilisée comme illustré dans la figure A.11. Le réseau BN est proposé dans A.4 en tant que solution de détection d'anomalies. Une sonde de surveillance MMT est couplée à chaque routeur pour extraire et collecter des données concernant les 18 mesures toutes les 5 secondes. Cependant, pour démontrer et évaluer toutes les fonctions de l'algorithme VE, nous considérons l'une des métriques comme inconnue ; sinon, la fonction *factor marginalisation* ne serait pas utilisée car il n'y aurait pas de variable à éliminer. L'approche proposée est implémentée en utilisant PiCN [152], qui est la dernière version de NFN, écrite en Python. Elle est comparée à un algorithme d'inférence du réseau bayésien standard exploitant une bibliothèque open-source nommée pgmpy [153] également écrite en Python.

Performances dans le temps

Pour évaluer les performances de notre approche, nous simulons un trafic normal pendant des intervalles de 10 minutes et nous mesurons l'évolution du temps de traitement, la mise en cache, le calcul local et les requêtes vers d'autres nœuds. Comme représenté sur la figure A.12.a, le cache est presque vide au début. L'efficacité de notre solution est montrée lorsque le cache est rempli par les résultats des exécutions précédentes ou de la propagation des nœuds voisins. Dans ce cas, le temps de calcul est inférieur à celui de l'approche standard. Ceci est confirmé par le résultat de la figure A.12.b, qui montre que le pourcentage moyen de calcul local et de requêtes vers d'autres nœuds est le plus élevé au début. De plus, nous notons que la partie des hits du cache augmente initialement avec le temps, puis se stabilise puisque le

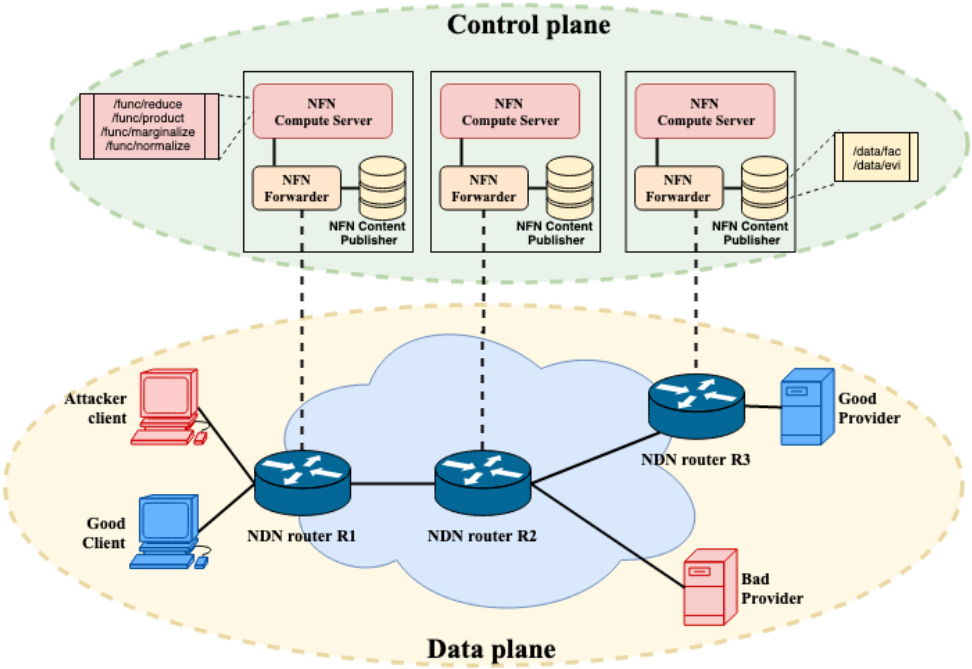


FIGURE A.11: Topologie de l'expérience

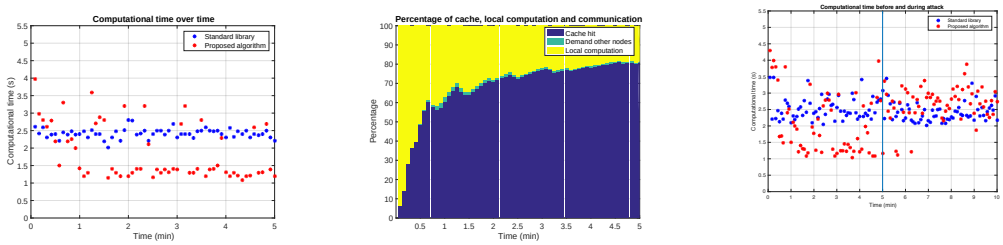


FIGURE A.12: Performance de l'approche proposée dans le temps: a) Instantané du temps de calcul dans le temps ; b) Évolution de la proportion des demandes au fil du temps ; (c) Instantané du temps de calcul avant et pendant l'attaque

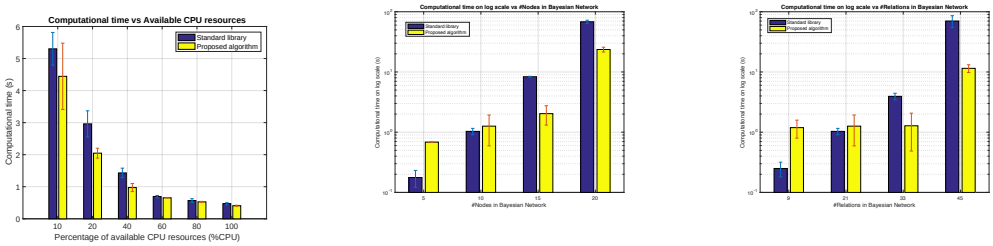


FIGURE A.13: Impacts sur le temps de calcul (échelle semi-logarithmique): (a) ressources CPU disponibles; (b) Nombre de nœuds dans BN; (c) Nombre de relations en BN

cache atteint sa limite. Le pourcentage des accès au cache après 10 minutes est de 80%, ce qui montre l'efficacité de notre approche.

Impact du trafic normal et anormal

Une autre évolution du temps de calcul, illustrée à la figure A.12.c, est évaluée dans le cas d'une attaque, où l'on simule un trafic normal pendant 5 minutes, puis

l'attaque CPA pendant 5 minutes. La ligne bleue verticale marque l'heure de début de l'attaque. Après l'attaque, le temps de calcul augmente. Étant donné que les métriques lors d'une attaque sont anormales par rapport au trafic normal, le résultat de la fonction de détection est introuvable dans le cache et nécessite du temps supplémentaire pour calculer les opérations dans l'algorithme d'inférence.

Impact des ressources CPU disponibles

Le résultat montré dans la figure A.13.a prouve que le temps de calcul dans notre approche est inférieur à celui de l'approche standard lorsque la capacité de calcul du nœud d'hébergement est limitée. Lorsque la ressource CPU disponible du routeur est inférieure à 0,4, l'approche proposée fonctionne mieux. Cela vient du fait que lorsque le routeur dispose de ressources de calcul suffisantes, il consomme toutes les ressources pour exécuter les opérations de l'algorithme d'inférence. Par conséquent, une approche standard peut consommer toutes les ressources et effectuer rapidement des opérations de calcul. En revanche, lorsqu'il atteint sa limite de calcul (en cas de ressources limitées), l'utilisation du cache permet de mieux exécuter les opérations de calcul que l'approche standard. Elle réduit le temps de calcul de 10% à 30 %.

Impact de la complexité du BN

Pour évaluer les performances de notre approche en fonction de la complexité de BN, le nombre de nœuds dans notre BN et le nombre de relations sont considérés. Comme représenté sur les figures A.13.b et A.13.c, lorsque le BN est simple, comptant ainsi un petit nombre de nœuds (moins de 10) ou de relations (moins de 21), le temps de calcul dans l'inférence standard est sans aucun doute meilleur que dans notre approche. La raison en est que le nombre d'opérations de calcul est limité, ce qui signifie qu'il n'a pas besoin de l'aide d'un cache ou d'autres nœuds pour effectuer les calculs. Cependant, lorsque le BN devient complexe, les performances de notre approche sont nettement meilleures que la méthode d'inférence standard. Dans ce cas, notre approche calcule 4 à 8 fois plus vite que celle standard. En fait, le nombre d'opérations de calcul est énorme et dépasse la capacité de calcul du nœud. En conséquence, l'utilisation d'un cache et l'aide d'autres nœuds deviennent extrêmement importantes. Les résultats montrent que nous bénéficions de l'infrastructure NFN lorsque le BN est complexe et nécessite un nombre important d'opérations.

A.7 Conclusions et perspectives

A.7.1 Conclusions

De nos jours, l'Internet a évolué loin de sa conception initiale qui, centrée sur l'hôte, montre ses inconvénients à l'ère des réseaux sociaux et du partage de contenu à grande échelle. Le paradigme de Information-Centric Networking est proposé comme une approche claire pour le futur Internet pour résoudre le problème. Named Data Networking est la proposition la plus aboutie des architectures ICN. Une telle nouvelle approche nécessite toutefois un effort important pour la déployer, la gérer et la sécuriser. Par conséquent, les solutions pour déployer NDN de manière sécurisée et efficace sont critiques pour les opérateurs de réseau qui voudraient mettre en oeuvre NDN à grande échelle. Dans un tel contexte, nous proposons dans cette thèse un plan de controle de la sécurité orienté contenu pour NDN.

Avant de déployer NDN, toutes les menaces potentielles doivent être identifiées et détectées. Parmi les attaques NDN, CPA est celle qui a le plus retenu l'attention de la communauté de recherche NDN, car il s'agit d'une attaque spécifique à ce paradigme de communication et qui est difficilement détectée par un détecteur standard (i.e. basé sur une seule métrique). En conséquence, nous avons examiné finements différentes approches pour détecter une telle attaque et nous avons constaté qu'elle a un impact sur divers composant d'un noeud NDN. Ainsi, nous avons sélectionné une approche par classifieur à base de réseau Bayesien (BNC) car elle peut corréler plusieurs événements et leurs impacts sur un ensemble de métriques pour classifier des comportements normaux et anormaux. En outre, BNC peut également naturellement gérer la nature aléatoire sous-jacente des métriques observées en utilisant l'approche probabiliste de Bayes, il est donc capable de gérer les dépendances entre les variables et les événements pertinents dans une attaque. Cependant, comme d'autres méthodes d'apprentissage automatique ou probabilistes, un BNC est coûteux en termes de consommation de ressources et de temps de calcul.

Nous avons abordé la première limitation en proposant un plan de surveillance de la sécurité pour NDN car il est indispensable avant tout déploiement potentiel de cette nouvelle architecture dans un contexte d'exploitation par tout fournisseur. Une liste de métriques est proposée pour couvrir le plan de données qui aide un détecteur à s'attaquer non seulement aux attaques actuelles dans NDN, mais également à toutes les attaques potentielles à l'avenir. Cependant, deux questions se posent pour le plan de surveillance proposé. Premièrement, les métriques couvrent différents aspects des composants NDN; par conséquent, leurs valeurs

sont également différentes des autres. Deuxièmement, les comportements anormaux et les attaques pourraient avoir un impact sur plusieurs métriques, mais certains changements dans la charge du réseau pourraient également modifier diverses métriques. Deux niveaux de détection sont proposés respectivement pour aborder ces problèmes. Tout d'abord, nous proposons un micro-détecteur pour détecter si une métrique dévie de son comportement attendu comme normal ; le microdétecteur dépend uniquement du comportement d'une métrique. Deuxièmement, nous proposons un moteur de corrélation utilisant un réseau bayésien qui corrèle les alertes de toutes les métriques et effectue l'inférence pour prédire si un routeur est attaqué. En utilisant CPA comme cas d'utilisation, nous collecterons des données à partir d'un réseau NDN testé pour évaluer le plan de surveillance que nous proposons. Les résultats numériques prouvent que même pour une petite charge utile d'attaque, le plan de surveillance peut détecter une attaque CPA avec une grande efficacité.

Le déploiement de NDN est une tâche stratégique et complexe, cela peut être rendu possible par des technologies innovantes telles que SDN, NFV et des solutions d'orchestration. Dans ce contexte, nous proposons une orchestration orientée contenu qui au delà du seul déploiement d'un réseau est capable d'orchestrer des fonctions de sécurité. Nous avons défini les éléments de base pour la conception et la mise en œuvre d'une orchestration orienté contenu pour les réseaux NDN virtualisés. Nous avons montré que la norme TOSCA peut être étendue pour devenir un modèle de spécification de services basés sur le contenu. Avec de nouveaux composants d'orchestration, nous avons démontré qu'un orchestrateur orienté contenu peut à la fois déployer automatiquement une topologie NDN virtuelle et appliquer une reconfiguration dynamique dans le réseau virtuel déclenchée par des politiques de sécurité et de montée en charge. Pour évaluer l'orchestrateur proposé, nous avons collecté différents indicateurs dans deux scénarios d'évaluation différentes pour ces deux tâches mentionnées ci-dessus. Les résultats numériques ont démontré que l'orchestrateur proposé peut à la fois déployer automatiquement une topologie NDN virtuelle, mais également appliquer une reconfiguration dynamique dans le réseau virtuel déclenchée par des politiques de sécurité et de montée en charge.

Enfin, nous avons proposé une architecture de détection d'anomalies orientée contenu. L'architecture de détection d'anomalies orientée contenu est motivée par notre observation qu'un plan de sécurité surveille en permanence un réseau à grande échelle; un grand nombre d'opérations de sécurité sont répétées. Nous avons défini les éléments de base pour la conception et la mise en œuvre d'un algorithme d'inférence de réseau bayésien pris en charge par NFN. Dans ce but, nous avons montré que l'algorithme VE dans BN peut être transformé en fonctions de λ-calculus

puis, grâce à NFN, mettre en cache et partager les résultats entre les nœuds. Dans le cadre de la détection CPA, nous avons collecté les résultats numériques pour démontrer les performances de l'algorithme proposé. Les résultats numériques prouvent que le BN de support NFN proposé fonctionne mieux le BN hérité dans le cas où la structure du BN est complexe et dans le cas de ressources de calcul limitées. En conséquence, le support NFN proposé pourrait apporter un bénéfice important pour des fonctions complexes de détection d'anomalies opérées dans des dispositifs à ressources de calcul limitées.

A.7.2 Perspectives

Le travail réalisé dans cette thèse ouvre plusieurs perspectives. Dans le cadre initial de cette thèse, le plan de surveillance proposé pourrait encore être amélioré. Par ailleurs, sur le long terme, le travail accompli dans cette thèse ouvre la voie vers un plan de contrôle orienté contenu qui peut être étendu pour être utilisé comme une application pour la détection d'anomalies en général.

Comme NDN est encore une architecture nouvelle et se trouve toujours dans un environnement académique et expérimental, de nouvelles attaques pourraient émerger lorsque l'architecture deviendra plus populaire. Au sein des solutions que nous avons élaborées, une liste de métriques est proposée pour couvrir tous les éléments des plans de données NDN. En conséquence, même s'il est question d'une nouvelle attaque, le plan de surveillance proposé pourrait également caractériser ces comportements. Ensuite, ces comportements seront utilisés pour entraîner le réseau bayésien, de sorte que le plan de surveillance proposé soit extensible pour d'autres attaques NDN. Cependant, les BNC proposés dépendent des comportements d'une attaque connue. Par conséquent, il faut du temps pour entraîner le détecteur d'anomalies lorsqu'une nouvelle attaque émerge. La phase d'apprentissage est un inconvénient de notre plan de surveillance proposé car une attaque «zero-day» pourrait l'exploiter. Plus précisément, chaque attaque connue est actuellement représentée comme une classe dans le nœud de sortie du BNC parmi la classe "normale" représentant le trafic normal. Pour surmonter le problème mentionné ci-dessus, nous pouvons ajouter un autre BNC avec la même structure, mais le nœud de sortie se compose uniquement de deux classes: «anormal» et «normal» pour les alertes si des comportements anormaux se sont produits dans le réseau sans identifier le type de attaque. En ce qui concerne l'inférence bayésienne, les travaux de cette thèse considèrent seulement l'algorithme VE comme l'algorithme d'inférence. Néanmoins, d'autres algorithmes d'inférence peuvent également se transformer en λ-calculus pour tirer parti des avantages du cache dans NFN. L'algorithme de Clique tree est un tel exemple, car son fonctionneemnt natif est proposé pour mettre en cache les calculs pendant le processus d'inférence.

L'objectif initial de l'approche proposée est de sécuriser le déploiement de NDN. Cependant, l'hypothèse selon laquelle les fonctions de sécurité répètent leurs opérations ne se limite pas uniquement au contexte de BNC et NDN. Par exemple, ce travail pourrait s'appliquer à différents scénarios, comme celui d'un détecteur d'anomalie de trafic réseau. Il ne fait aucun doute que le trafic d'un serveur en "heures creuses" ne change pas beaucoup, et par conséquent, le comportement des fonctions de sécurité se voie répété pendant ces "heures creuses". Un autre exemple est celui des caméras de détection d'intrusion.

Dans ce contexte, lorsqu'il n'y a pas de comportement suspect, la sortie de la caméra reste inchangée. Dans les deux cas, comme l'entrée pour les fonctions de sécurité ne change pas, nous pouvons bénéficier des propriétés de cache. En ce qui concerne l'algorithme des fonctions de sécurité, un bon exemple est le deep learning, où la détection est basée sur des réseaux de neurones multicouches. En utilisant NFN, le calcul pour ces différentes couches pourrait être distribué et mis en cache sur le réseau. Le travail accompli dans cette thèse est conçu et développé dans un scénario et déploiement réel et ciblé, mais il ouvre la voie vers la conception de plans de contrôle orientés contenu pouvant être utilisés dans des scénario et environnements variés. Cette capacité ouvre une nouvelle porte pour l'approche proposée à utiliser dans NDN et dans d'autres architectures de réseau, en particulier, dans les situations où l'écart entre les fonctions de sécurité et les ressources contraintes des nœuds d'exécution est important.

Publications

International Journals

[122] Nguyen, T., **Mai, H. L.**, Doyen, G., Cogranne, R., Mallouli, W., de Oca, E. M., Festor, O. (2018). *A security monitoring plane for named data networking deployment*. IEEE Communications Magazine, 56(11), 88-94.

[157] Nguyen, T., **Mai, H. L.**, Cogranne, R., Doyen, G., Mallouli, W., Nguyen, L., ... Festor, O. (2019). *Reliable Detection of Interest Flooding Attack in Real Deployment of Named Data Networking*. IEEE Transactions on Information Forensics and Security, 14(9), 2470-2485.

[158] Doyen, G., Cholez, T., Mallouli, W., Mathieu, B., **Mai, H.L.**, Marchal, X., Kondo, D., Aouadj, M., Ploix, A., Montes-de-Oca, E. and Foster, O., 2019. *An Orchestrated NDN Virtual Infrastructure Transporting Web Traffic: Design, Implementation, and First Experiments with Real End Users*. IEEE Communications Magazine, 57(6), pp.33-39.

International Conferences

[45] **Mai, H. L.**, Nguyen, N. T., Doyen, G., Ploix, A., Cogranne, R. (2016, June). *On the readiness of NDN for a secure deployment: The case of pending interest table*. In IFIP International Conference on Autonomous Infrastructure, Management and Security (AIMS) (pp. 98-110). Springer, Cham.

[127] **Mai, H. L.**, Nguyen, T., Doyen, G., Cogranne, R., Mallouli, W., de Oca, E. M., Festor, O. (2018, April). *Towards a security monitoring plane for named data networking and its application against content poisoning attack*. In IEEE/IFIP Network Operations and Management Symposium (NOMS) (pp. 1-9). IEEE.

[130] **Mai, H. L.**, Doyen, G., Mallouli, W., de Oca, E. M., Festor, O. (2019, October). *Towards Content-Centric Control Plane Supporting Efficient Anomaly Detection Functions*. In IEEE/IFIP International Conference on Network and Service Management (CNSM). IEEE.

[159] **Mai, H. L.**, Aouadj, M., Doyen, G., Mallouli, W., de Oca, E. M., Festor, O. (2019, April). *Toward Content-Oriented Orchestration: SDN and NFV as Enabling*

Technologies for NDN. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 594-598). IEEE.

[155] **Mai, H.L.**, Aouadj, M., Doyen, G., Kondo, D., Marchal, X., Cholez, T., De Oca, E.M. and Mallouli, W., 2018, February. *Implementation of content poisoning attack detection and reaction in virtualized NDN networks*. In 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN) (pp. 1-3). IEEE.

Book Chapter

[160] Liyanage, M., Ahmad, I., Okwuibe, J., de Oca, E. M., **Mai, H. L.**, López, O., Uriarte, M. (2018). *Software Defined Security Monitoring in 5G Networks. A Comprehensive Guide to 5G Security*; John Wiley Sons: Hoboken, NJ, USA, 231.

Bibliography

[1] B. Rainer and S. Petscharnig, "Challenges and opportunities of named data networking in vehicle-to-everything communication: A review," *Information*, vol. 9, no. 11, p. 264, 2018.

[2] B. Quinn and K. Almeroth, "Ip multicast applications: Challenges and solutions," 2001.

[3] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE network*, vol. 14, no. 1, pp. 78–88, 2000.

[4] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[5] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim *et al.*, "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and tutorials*, vol. 7, no. 1-4, pp. 72–93, 2005.

[6] S. Ratnasamy, I. Stoica, and S. Shenker, "Routing algorithms for dhts: Some open questions," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 45–52.

[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[8] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in Content-oriented Architectures," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018586

[9] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[11] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 37. ACM, 2007, pp. 181–192.

[12] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of Information (NetInf)–An information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[13] Scalable and adaptive internet solutions (sail) project. [Online]. Available: http://www.sail-project.eu/

[14] N. Fotiou, P. Nikander, D. Trossen, G. C. Polyzos, and others, "Developing Information Networking Further: From PSIRP to PURSUIT." in *Broadnets*. Springer, 2010, pp. 1–13.

[15] S. Tarkoma, M. Ain, and K. Visala, "The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture." in *Future Internet Assembly*, 2009, pp. 102–111.

[16] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[18] W. Stallings, "Software-defined networks and openflow," *The internet protocol Journal*, vol. 16, no. 1, pp. 2–14, 2013.

[19] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[20] A. Rahman, D. Trossen, D. Kutscher, and R. Ravindran, "Deployment considerations for information-centric networking (icn)," *ICNRG draft, available at https://tools.ietf.org/html/draft-irtf-icnrg-deployment-guidelines-04*, 2018.

[21] 3GPP, ""technical specification group services and system aspects; system architecture for the 5g system (rel.15)," 2017.

[22] M. Vahlenkamp, F. Schneider, D. Kutscher, and J. Seedorf, "Enabling information centric networking in ip networks using sdn," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013, pp. 1–6.

[23] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "Conet: a content centric inter-networking architecture," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2011, pp. 50–55.

[24] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over sdn and openflow: Architectural aspects and experiments on the ofelia testbed," *Computer Networks*, vol. 57, no. 16, pp. 3207–3221, 2013.

[25] Ofelia project. [Online]. Available: http://www.fp7-ofelia.eu/

[26] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer issues in information centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012, pp. 19–24.

[27] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti, "Supporting information-centric functionality in software defined networks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 6645–6650.

[28] N. L. M. van Adrichem and F. A. Kuipers, "Ndnflow: Software-defined named data networking," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[29] X. N. Nguyen, D. Saucez, and T. Turletti, "Efficient caching in content-centric networks using openflow," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 1–2.

[30] ——, "Efficient caching in content-centric networks using openflow," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 67–68.

[31] Virtualized icn. [Online]. Available: https://wiki.fd.io/view/Vicn

[32] M. Sardara, L. Muscariello, J. Augé, M. Enguehard, A. Compagno, and G. Carofiglio, "Virtualized icn (vicn): Towards a unified network virtualization framework for icn experimentation," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ser. ICN '17. New

York, NY, USA: ACM, 2017, pp. 109–115. [Online]. Available: http://doi.acm.org/10.1145/3125719.3125726

[33] European telecommunications standards institute, industry specification groups ? nfv. [Online]. Available: https://www.etsi.org/technologies-clusters/technologies/nfv

[34] B. Mathieu, G. Doyen, W. Mallouli, T. Silverston, O. Bettan, F. X. Aguessy, T. Cholez, A. Lahmadi, P. Truong, and E. M. d. Oca, "Monitoring and securing new functions deployed in a virtualized networking environment," in *2015 10th International Conference on Availability, Reliability and Security*, Aug 2015, pp. 741–748.

[35] X. Marchal, M. E. Aoun, B. Mathieu, W. Mallouli, T. Cholez, G. Doyen, P. Truong, A. Ploix, and E. M. De Oca, "A virtualized and monitored ndn infrastructure featuring a ndn/http gateway," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN '16. New York, NY, USA: ACM, 2016, pp. 225–226. [Online]. Available: http://doi.acm.org/10.1145/2984356.2985238

[36] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A survey of security attacks in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1441–1454, 2015.

[37] N. Kumar, A. K. Singh, A. Aleem, and S. Srivastava, "Security attacks in named data networking: A review and research directions," *Journal of Computer Science and Technology*, vol. 34, no. 6, pp. 1319–1350, 2019.

[38] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *2013 IFIP Networking Conference*. IEEE, 2013, pp. 1–9.

[39] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2426–2434.

[40] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache Privacy in Named-Data Networking," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, Jul. 2013, pp. 41–51.

[41] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim, "Protecting access privacy of cached contents in information centric networks," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 173–178.

[42] A. Mohaisen, H. Mekky, X. Zhang, H. Xie, and Y. Kim, "Timing Attacks on Access Privacy in Information Centric Networks and Countermeasures," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 675–687, Nov. 2015.

[43] Z. Xu, B. Chen, N. Wang, Y. Zhang, and Z. Li, "Elda: Towards efficient and lightweight detection of cache pollution attacks in ndn," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*. IEEE, 2015, pp. 82–90.

[44] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3. IEEE, 2000, pp. 2275–2280.

[45] H. L. Mai, N. T. Nguyen, G. Doyen, A. Ploix, and R. Cogranne, "On the readiness of ndn for a secure deployment: The case of pending interest table," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2016, pp. 98–110.

[46] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos and ddos in named data networking," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2013, pp. 1–7.

[47] H. Dai, Y. Wang, J. Fan, and B. Liu, "Mitigate ddos attacks in ndn by interest traceback," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 381–386.

[48] H. Salah and T. Strufe, "Evaluating and mitigating a collusive version of the interest flooding attack in ndn," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 938–945.

[49] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding ddos attacks in named data networking," in *38th annual IEEE conference on local computer networks*. IEEE, 2013, pp. 630–638.

[50] T. N. Nguyen, R. Cogranne, G. Doyen, and F. Retraint, "Detection of interest flooding attacks in named data networking using hypothesis testing," in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[51] Y. Xin, Y. Li, W. Wang, W. Li, and X. Chen, "A novel interest flooding attacks detection and countermeasure scheme in ndn," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[52] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[53] K. Wang, H. Zhou, Y. Qin, and H. Zhang, "Cooperative-filter: countering interest flooding attacks in named data networking," *Soft Computing*, vol. 18, no. 9, pp. 1803–1813, 2014.

[54] K. Ding, Y. Liu, H.-H. Cho, H.-C. Chao, and T. K. Shih, "Cooperative detection and protection for interest flooding attacks in named data networking," *International Journal of Communication Systems*, vol. 29, no. 13, pp. 1968–1980, 2016.

[55] A. Karami and M. Guerrero-Zapata, "A hybrid multiobjective rbf-pso method for mitigating dos attacks in named data networking," *Neurocomputing*, vol. 151, pp. 1262–1282, 2015.

[56] Z. Li and J. Bi, "Interest cash: an application-based countermeasure against interest flooding for dynamic content in named data networking," in *Proceedings of The Ninth International Conference on Future Internet Technologies*, 2014, pp. 1–6.

[57] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM computer communication review*, vol. 42, no. 3, pp. 62–67, 2012.

[58] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[59] N. Ntuli and S. Han, "Detecting router cache snooping in named data networking," in *2012 International Conference on ICT Convergence (ICTC)*. IEEE, 2012, pp. 714–718.

[60] M. Gao, X. Zhu, and Y. Su, "Protecting router cache privacy in named data networking," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2015, pp. 1–5.

[61] E. Dogruluk, A. Costa, and J. Macedo, "Evaluating privacy attacks in named data network," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 1251–1256.

[62] A. Chaabane, E. De Cristofaro, M. A. Kaafar, and E. Uzun, "Privacy in content-oriented networking: Threats and countermeasures," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 25–33, 2013.

[63] A. Mohaisen, X. Zhang, M. Schuchard, H. Xie, and Y. Kim, "Protecting access privacy of cached contents in information centric networks," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 173–178.

[64] N. Abani and M. Gerla, "Centrality-based caching for privacy in information-centric networks," in *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016, pp. 1249–1254.

[65] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda, "Privacy implications of ubiquitous caching in named data networking architectures," *Technical Report TR-iSecLab-0812-001, ISecLab, Tech. Rep.*, 2012.

[66] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in named data networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013.

[67] H. Guo, X. Wang, K. Chang, and Y. Tian, "Exploiting path diversity for thwarting pollution attacks in named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2077–2090, 2016.

[68] W. Gilks, S. Richardson, and D. Spieglhalter, "Markov chain monte carlo in practice, 520 pp," 1996.

[69] A. Karami and M. Guerrero-Zapata, "An anfis-based cache replacement method for mitigating cache pollution attacks in named data networking," *Computer Networks*, vol. 80, pp. 51–65, 2015.

[70] H. Park, I. Widjaja, and H. Lee, "Detection of cache pollution attacks using randomness checks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 1096–1100.

[71] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a Haystack: Mitigating Content Poisoning in Named-Data Networking," in *Proceedings of NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[72] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in LRU caching?" *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 59–67, 2013.

[73] D. Kim, S. Nam, J. Bi, and I. Yeom, "Efficient Content Verification in Named Data Networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 109–116.

[74] I. Ribeiro, A. Rocha, C. Albuquerque, and F. Guimaraes, "On the possibility of mitigating content pollution in Content-Centric Networking," in *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*. IEEE, 2014, pp. 498–501.

[75] C. Ghali, G. Tsudik, and E. Uzun, "Network-Layer Trust in Named-Data Networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 12–19, 2014.

[76] A. Karami and M. Guerrero-Zapata, "An anfis-based cache replacement method for mitigating cache pollution attacks in named data networking," *Computer Networks*, vol. 80, pp. 51–65, 2015.

[77] S. DiBenedetto and C. Papadopoulos, "Mitigating Poisoned Content with Forwarding Strategy," in *The third Workshop on Name-Oriented Mobility (NOM)*. San Francisco, USA: IEEE, 2016.

[78] C. Ghali, G. Tsudik, E. Uzun *et al.*, "Needle in a haystack: Mitigating content poisoning in named-data networking," in *Proceedings of NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[79] X. Hu, J. Gong, G. Cheng, G. Zhang, and C. Fan, "Mitigating content poisoning with name-key based forwarding and multipath forwarding based inband probe for energy management in smart cities," *IEEE Access*, vol. 6, pp. 39 692–39 704, 2018.

[80] S. R. Gunn *et al.*, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998.

[81] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[82] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[83] F. V. Jensen *et al.*, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.

[84] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[85] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[86] H. Zhang, "Exploring conditions for the optimality of naive bayes," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 02, pp. 183–198, 2005.

[87] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using bayesian networks for cyber security analysis," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2010, pp. 211–220.

[88] F. Fooladvandi, C. Brax, P. Gustavsson, and M. Fredin, "Signature-based activity detection based on bayesian networks acquired from expert knowledge," in *2009 12th International Conference on Information Fusion*. IEEE, 2009, pp. 436–443.

[89] E. T. Axelrad, P. J. Sticha, O. Brdiczka, and J. Shen, "A bayesian network model for predicting insider threats," in *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 82–89.

[90] S. Yang and K.-C. Chang, "Comparison of score metrics for bayesian network learning," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 32, no. 3, pp. 419–428, 2002.

[91] H. Amirkhani, M. Rahmati, P. J. Lucas, and A. Hommersom, "Exploiting experts' knowledge for structure learning of bayesian networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2154–2170, 2016.

[92] L. M. d. Campos, "A scoring function for learning bayesian networks based on mutual information and conditional independence tests," *Journal of Machine Learning Research*, vol. 7, no. Oct, pp. 2149–2187, 2006.

[93] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, prediction, and search*. MIT press, 2000.

[94] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3741–3782, 2014.

[95] M. Scutari, C. E. Graafland, and J. M. Gutiérrez, "Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms," *International Journal of Approximate Reasoning*, vol. 115, pp. 235–253, 2019.

[96] D. Margaritis, "Learning bayesian network model structure from data," Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, Tech. Rep., 2003.

[97] S. Yaramakala and D. Margaritis, "Speculative markov blanket discovery for optimal feature selection," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 4–pp.

[98] A. Stuart, S. Arnold, J. K. Ord, A. O'Hagan, and J. Forster, *Kendall's advanced theory of statistics*. Wiley, 1994.

[99] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial intelligence*, vol. 42, no. 2-3, pp. 393–405, 1990.

[100] P. Dagum and M. Luby, "Approximating probabilistic inference in bayesian belief networks is np-hard," *Artificial intelligence*, vol. 60, no. 1, pp. 141–153, 1993.

[101] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 50, no. 2, pp. 157–194, 1988.

[102] F. V. Jensen, K. G. Olesen, and S. K. Andersen, "An algebra of bayesian belief universes for knowledge-based systems," *Networks*, vol. 20, no. 5, pp. 637–659, 1990.

[103] S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in ak-tree," *SIAM Journal on Algebraic Discrete Methods*, vol. 8, no. 2, pp. 277–284, 1987.

[104] F. V. Jensen and F. Jensen, "Optimal junction trees," in *Uncertainty Proceedings 1994*. Elsevier, 1994, pp. 360–366.

[105] K. Shoikhet and D. Geiger, "A practical algorithm for finding optimal triangulations," in *AAAI/IAAI*, 1997, pp. 185–190.

[106] A. Becker and D. Geiger, "A sufficiently fast algorithm for finding close to optimal clique trees," *Artificial Intelligence*, vol. 125, no. 1-2, pp. 3–17, 2001.

[107] L. Zheng, O. Mengshoel, and J. Chong, "Belief propagation by message passing in junction trees: Computing each message faster using gpu parallelization," *arXiv preprint arXiv:1202.3777*, 2012.

[108] L. Zheng and O. Mengshoel, "Optimizing parallel belief propagation in junction treesusing regression," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 757–765.

[109] L. Zheng and O. J. Mengshoel, "Exploring multiple dimensions of parallelism in junction tree message passing," in *Proceedings of the 2013 UAI Conference on Application Workshops: Big Data meet Complex Models and Models for Spatial, Temporal and Network Data-Volume 1024*. Citeseer, 2013, pp. 87–96.

[110] Y. Xia and V. K. Prasanna, "Distributed evidence propagation in junction trees on clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1169–1177, 2011.

[111] N. Ma, Y. Xia, and V. K. Prasanna, "Task parallel implementation of belief propagation in factor graphs," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 2012, pp. 1944–1953.

[112] ——, "Parallel exact inference on multicore using mapreduce," in *2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing*. IEEE, 2012, pp. 187–194.

[113] Y. Xia and V. K. Prasanna, "Parallel evidence propagation on multicore processors," *The Journal of Supercomputing*, vol. 57, no. 2, pp. 189–202, 2011.

[114] ——, "Self-adaptive evidence propagation on manycore processors," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 1407–1416.

[115] R. Fung and K.-C. Chang, "Weighing and integrating evidence for stochastic simulation in bayesian networks," in *Machine Intelligence and Pattern Recognition*. Elsevier, 1990, vol. 10, pp. 209–219.

[116] D. Gamerman and H. F. Lopes, *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[117] A. E. Gelfand, S. E. Hills, A. Racine-Poon, and A. F. Smith, "Illustration of bayesian inference in normal data models using gibbs sampling," *Journal of the American Statistical Association*, vol. 85, no. 412, pp. 972–985, 1990.

[118] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[119] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[120] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[121] H. L. Mai, N. Tan, D. Guillaume, R. Cogranne, M. Wissam, M.-d.-O. Edgardo, and F. Olivier, "Towards a security monitoring plane for named data networking and its application against content poisoning attack," in *to appear In IEEE/IFIP Network Operations and Management Symposium (NOMS), 2018*. IEEE, 2018.

[122] T. Nguyen, H.-L. Mai, G. Doyen, R. Cogranne, W. Mallouli, E. M. De Oca, and O. Festor, "A security monitoring plane for named data networking deployment," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 88–94, 2018.

[123] S. E. Yusuf, J. B. Hong, M. Ge, and D. S. Kim, "Composite metrics for network security analysis," *Software Networking*, vol. 2018, no. 1, pp. 137–160, 2018.

[124] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto, and others, "NFD developer?s guide," Technical Report NDN-0021, Oct. 2016. [Online]. Available: https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf

[125] T. Nguyen, X. Marchal, G. Doyen, T. Cholez, and R. Cogranne, "Content Poisoning in Named Data Networking: Comprehensive characterization of real deployment," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 72–80.

[126] D. Palma, M. Rutkowski, and T. Spatzier, "Tosca simple profile in yaml version 1.0," *OASIS Committee Specification*, vol. 1, 2016.

[127] H. L. Mai, N. T. Nguyen, G. Doyen, R. Cogranne, M. Wissam, E. Montes de Oca, and O. Festor, "Towards a security monitoring plane for named data networking: Application to content poisoning attack," in *Network Operations and Management Symposium (NOMS), 2018 IEEE*. IEEE, 2018.

[128] D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Name anomaly detection for icn," in *Local and Metropolitan Area Networks (LANMAN), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1–6.

[129] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[130] H. L. Mai, , G. Doyen, M. Wissam, E. Montes de Oca, and O. Festor, "Towards content-centric control plane supporting efficient anomaly detection functions," in *Network Operations and Management Symposium (NOMS), 2018 IEEE*. IEEE, 2018.

[131] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[132] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, "An information centric network for computing the distribution of computations," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 137–146.

[133] C. Tschudin and M. Sifalakis, "Named functions and cached computations," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*. IEEE, 2014, pp. 851–857.

[134] J.-L. Krivine, "A call-by-name lambda-calculus machine," *Higher-Order and Symbolic Computation*, vol. 20, no. 3, pp. 199–207, 2007.

[135] C. Scherb, B. Faludi, and C. Tschudin, "Execution state management in named function networking," in *Workshop on Information-Centric Fog Computing, IFIP Networking*, 2017.

[136] M. Król and I. Psaras, "Nfaas: named function as a service," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 134–144.

[137] C. Tschudin and M. Sifalakis, "Named functions for media delivery orchestration," in *2013 20th International Packet Video Workshop*. IEEE, 2013, pp. 1–8.

[138] C. Scherb, U. Schnurrenberger, C. Marxer, and C. Tschudin, "In-network live stream processing with named functions," in *Workshop on Information-Centric Fog Computing, IFIP Networking*, 2017.

[139] C. Marxer and C. Tschudin, "Improved content addressability through relational data modeling and in-network processing elements," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 29–35.

[140] C. Marxer, C. Scherb, and C. F. Tschudin, "Access-controlled in-network processing of named data." in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 77–82.

[141] C. Marxer and C. Tschudin, "Schematized access control for data cubes and trees," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 170–175.

[142] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2017.

[143] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proceedings of the Workshop on Mobile Edge Communications*. ACM, 2017, pp. 7–12.

[144] J. Burke, "Browsing an augmented reality with named data networking," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017, pp. 1–9.

[145] P. TalebiFard, R. Ravindran, A. Chakraborti, J. Pan, A. Mercian, G. Wang, and V. C. Leung, "An information centric networking approach towards contextualized edge service," in *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*. IEEE, 2015, pp. 250–255.

[146] Q. Wang, B. Lee, N. Murray, and Y. Qiao, "Cs-man: Computation service management for iot in-network processing," in *Signals and Systems Conference (ISSC), 2016 27th Irish*. IEEE, 2016, pp. 1–6.

[147] Y. Ye, Y. Qiao, B. Lee, and N. Murray, "Piot: Programmable iot using information centric networking," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 825–829.

[148] Q. Wang, B. Lee, N. Murray, and Y. Qiao, "Iproiot: An in-network processing framework for iot using information centric networking," in *Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on*. IEEE, 2017, pp. 93–98.

[149] H. Zhang, Z. Wang, C. Scherb, C. Marxer, J. Burke, L. Zhang, and C. F. Tschudin, "Sharing mhealth data via named data networking." in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 142–147.

[150] L. M. JSM, V. Lokesh, and G. C. Polyzos, "Energy efficient context based forwarding strategy in named data networking of things," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM, 2016, pp. 249–254.

[151] D. Mansour and C. F. Tschudin, "Towards a monitoring protocol over information-centric networks." in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 60–64.

[152] "Picn: Python icn and nfn by university of basel," https://github.com/cn-uofbasel/PiCN.

[153] "Python library for probabilistic graphical models," https://github.com/pgmpy/pgmpy.

[154] T. N. Nguyen, R. Cogranne, G. Doyen, and F. Retraint, "Detection of Interest flooding attacks in named data networking using hypothesis testing," in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[155] H. L. Mai, M. Aouadj, G. Doyen, D. Kondo, X. Marchal, T. Cholez, E. M. de Oca, and W. Mallouli, "Implementation of content poisoning attack detection and reaction in virtualized ndn networks," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2018, pp. 1–3.

[156] T. Nguyen, X. Marchal, G. Doyen, T. Cholez, and R. Cogranne, "Content poisoning in named data networking: Comprehensive characterization of real deployment," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 72–80.

[157] T. Nguyen, H.-L. Mai, R. Cogranne, G. Doyen, W. Mallouli, L. Nguyen, M. El Aoun, E. M. De Oca, and O. Festor, "Reliable detection of interest flooding attack in real deployment of named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2470–2485, 2019.

[158] G. Doyen, T. Cholez, W. Mallouli, B. Mathieu, H.-L. Mai, X. Marchal, D. Kondo, M. Aouadj, A. Ploix, E. Montes-de Oca *et al.*, "An orchestrated ndn virtual infrastructure transporting web traffic: Design, implementation, and first experiments with real end users," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 33–39, 2019.

[159] H. L. Mai, M. Aouadj, G. Doyen, W. Mallouli, E. M. de Oca, and O. Festor, "Toward content-oriented orchestration: Sdn and nfv as enabling technologies for ndn," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 594–598.

[160] M. Liyanage, I. Ahmad, A. B. Abro, A. Gurtov, and M. Ylianttila, *Comprehensive Guide to 5G Security*. Wiley Online Library, 2018.

Hoang Long MAI

Doctorat : Systèmes SocioTechniques

Année 2020

Sécurité orientée contenu pour les réseaux NDN - application aux attaques par empoisonnement de type CPA

Internet a évolué et s'est éloigné de sa conception initiale. Le paradigme de réseau orienté contenu (ICN) est proposé comme une approche pour résoudre ce problème et *Named Data Networking* (NDN) est la proposition la plus mature des architectures ICN. Cependant, elle nécessite encore un effort considérable pour la déployer, la gérer et la sécuriser. Dans cette thèse, nous proposons un plan de sécurité orienté contenu pour les réseaux NDN et avons choisi l'attaque par empoisonnement de contenu CPA comme cas d'utilisation pour évaluer notre proposition. Premièrement, nous avons proposé un micro-détecteur pour déceler les déviations de métriques par rapport à leur comportement normal et un moteur de corrélation utilisant les réseaux bayésiens afin de combiner les micro-alertes et détecter l'occurrence d'attaques. Deuxièmement, nous avons proposé une orchestration orientée contenu pour la sécurité de NDN. Nous avons défini un profil TOSCA en étendant les nœuds de base pour ajouter des propriétés NDN et avons ajouté des politiques de sécurité NDN pour répondre aux incidents de sécurité ou de performance. Enfin, nous avons proposé un plan de contrôle de sécurité orienté contenu pour surveiller NDN. Plus spécifiquement, les fonctions de sécurité peuvent être distribuées dans Named Function Networking qui est un réseau orienté contenu et permet de répartir la tâche de calcul dans le réseau. Nous utilisons la détection d'anomalie Bayesienne comme cas d'utilisation pour démontrer que l'approche proposée peut améliorer les performances.

Mots clés : réseaux d'ordinateurs – détection des anomalies (informatique) – statistique bayésienne – systèmes virtuels (informatique).

Towards a Content-oriented Security Plane for Named Data Networking: Application to Content Poisoning Attack

Nowadays, the Internet has evolved far from its initial design. Motivated by growing change requirements to its core, the Information- Centric Networking (ICN) paradigm is proposed as a clean-slate approach for the Future Internet to address this issue and Named Data Networking (NDN) is the most mature proposal of ICN architectures. However, a novel approach also requires a significant effort to deploy, manage, and secure it. In this thesis, we proposed a content-oriented security plane for NDN and chose Content Poisoning Attack as the use-case to evaluate our proposal. In this context, firstly, we have proposed a micro detector which indicates if a metric is shifted from its normal behavior. Moreover, we also proposed a correlation engine using a Bayesian Network that gathers the alerts from all metrics and performs an inference to predict if a router is under attack. Secondly, we have proposed a content-oriented orchestration for the security of NDN. For this purpose, we defined a TOSCA profil for the NDN network by extending base nodes to add NDN properties. Moreover, typical NDN security policies are added to TOSCA policies for security or performance incidents. Finally, we propose a content-oriented security plan to monitor NDN. More specifically, security functions can be distributed in Named Function Networking (NFN), which is a content-oriented network that allows distributing the computation task in the network. We use the Bayesian anomaly detection as a use-case to demonstrate that the proposed approach can improve the performance.

Keywords: computer networks – anomaly detection (computer security) – bayesian statistical decision theory – virtual computer systems.

Thèse réalisée en partenariat entre :

anrt association nationale recherche technologie — Cifre Conventions — montimage — utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences pour l'Ingénieur"