Vítor António Gonçalves Ribeiro da Cunha

Mecanismos Dinâmicos de Segurança para Redes Softwarizadas e Virtualizadas

Dynamic Security Mechanisms for Softwarized and Virtualized Networks

Universidade de Aveiro
2022

Vítor António Gonçalves Ribeiro da Cunha

Mecanismos Dinâmicos de Segurança para Redes Softwarizadas e Virtualizadas

Dynamic Security Mechanisms for Softwarized and Virtualized Networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica do Doutor João Paulo Barraca, Professor auxiliar do Departamento Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Daniel Nunes Corujo, Professor auxiliar do Departamento Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020-UIDP/50008/2020.

This work was supported in part by the EU H2020 5GROWTH Project under Grant no. 856709.

o júri / the jury

presidente / president

Prof. Doutora Maria Adelaide de Pinho Almeida
professora catedrática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutora Pétia Georgieva
professora associada da Universidade de Aveiro

Prof. Doutor João Paulo Barraca
professor auxiliar da Universidade de Aveiro

Prof. Doutora Maria Solange Pires Ferreira Rito Lima
professora auxiliar da Universidade do Minho

Prof. Doutor Bruno Miguel de Oliveira Sousa
professor auxiliar da Universidade de Coimbra

Prof. Doutor José André Rocha Sá Moura
professor auxiliar do ISCTE - Instituto Universitário de Lisboa

agradecimentos /
acknowledgements

Agradeço à minha equipa de orientação (Prof. Dr. João Paulo Barraca e Prof. Dr. Daniel Corujo), à Universidade de Aveiro, e especialmente ao Instituto de Telecomunicações (Aveiro) por criarem um ambiente académico excecional para a realização desta tese. Agradeço também ao consórcio do H2020 5Growth, em particular aos participantes dos pilotos de Aveiro, com destaque para a Altice Labs, Instituto de Telecomunicações, e EFACEC, os quais foram excecionais na abertura e tempo despendido, e determinantes para a formulação de um ambiente realista de testes que comportava os mecanismos aqui propostos. Aproveito também para agradecer ao Prof. Dr. Rui Aguiar pelo extraordinário apoio durante esta longa jornada. Saudações para a equipa técnica do Instituto de Telecomunicações (Aveiro) e os meus colegas do grupo de investigação pelo seu incansável apoio e feedback sobre variadíssimas questões.

Palavras Chave

cibersegurança, moving target defense, software-defined networking, network function virtualization.

Resumo

A relação entre atacantes e defensores tem sido tradicionalmente assimétrica, com os atacantes a terem o tempo como vantagem para conceberem uma exploração que comprometa o defensor. O impulso para a Cloudificação do mundo torna a situação mais desafiante, pois reduz o custo de um ataque, com uma padronização *de facto* sobre um conjunto de protocolos. A descoberta de uma vulnerabilidade tem agora um impacto mais amplo em várias verticais (casos de uso empresarial), enquanto anteriormente, alguns estavam numa pilha de protocolos segregados que exigiam uma investigação independente das suas vulnerabilidades. Além disso, a definição de um perímetro dentro de um sistema Cloud não é trivial, enquanto antes, o equipamento dedicado já criava um perímetro. Esta proposta toma as mais recentes tecnologias de softwarização e virtualização da rede, ambas facilitadoras da Cloud, para criar novos mecanismos dinâmicos de segurança que incidem sobre esta relação assimétrica utilizando novas abordagens de Moving Target Defense (MTD). A utilização eficaz do espaço de exploração, combinada com as capacidades de reconfiguração de frameworks como Network Function Virtualization (NFV) e Management and Orchestration (MANO), deverá permitir ajustar dinamicamente os níveis de defesa para alcançar a segurança necessária, tal como definida pelo risco actualmente aceitável. As tarefas de optimização e de integração desta tese exploram estes conceitos. Além disso, os novos mecanismos propostos foram avaliados em casos de utilização no mundo real, tais como redes 5G ou outras infraestruturas de Network Slicing.

Keywords cybersecurity, moving target defense, software-defined networking, network function virtualization.

Abstract

The relationship between attackers and defenders has traditionally been asymmetric, with attackers having time as an upper hand to devise an exploit that compromises the defender. The push towards the Cloudification of the world makes matters more challenging, as it lowers the cost of an attack, with a *de facto* standardization on a set of protocols. The discovery of a vulnerability now has a broader impact on various verticals (business use cases), while previously, some were in a segregated protocol stack requiring independent vulnerability research. Furthermore, defining a perimeter within a cloudified system is non-trivial, whereas before, the dedicated equipment already created a perimeter. This proposal takes the newer technologies of network softwarization and virtualization, both Cloud-enablers, to create new dynamic security mechanisms that address this asymmetric relationship using novel Moving Target Defense (MTD) approaches. The effective use of the exploration space, combined with the reconfiguration capabilities of frameworks like Network Function Virtualization (NFV) and Management and Orchestration (MANO), should allow for adjusting defense levels dynamically to achieve the required security as defined by the currently acceptable risk. The optimization tasks and integration tasks of this thesis explore these concepts. Furthermore, the proposed novel mechanisms were evaluated in real-world use cases, such as 5G networks or other Network Slicing enabled infrastructures.

Contents

List of Figures

List of Tables

Glossary

2FA	Two Factor Authentication
5Gr-VS	5Growth Vertical Slicer
ASLR	Address Space Layout Randomization
BSS	Business Support System
C2	Command and Control
CAPEC	Common Attack Pattern Enumeration and Classification
CAPEX	Capital Expenditure
CBA	Cost-Benefit Analysis
CEA	Cost-Effectiveness Analysis
CIA	Confidentiality, Integrity, and Availability
CKC	Cyber Kill Chain
CMD	Cyber Mimic Defense
COTS	Consumer Off The Shelf
CPE	Customer Premises Equipment
CSMF	Communication Service Management Function
DESIR	Decoy-Enhanced Seamless Network IP Randomization
DHC	Double Hopping Communication
DHR	Dynamic Heterogeneous Redundancy
DoS	Denial of Service
DYNAT	Dynamic Network Address Translation
E2E	End-to-End
eMBB	enhanced Mobile Broadband
EPRM	End Point Route Mutation
ETSI	European Telecommunications Standards Institute
FPH	Fingerprint Hopping
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
IDS	Intrusion Detection System
IPS	Intrusion Protection System
IoT	Internet of Things
ISR	Instruction Set Randomization
KHSS	Keyed-Hashing based Self-Synchronization
KPI	Key Performance Indicator
LAN	Local Area Network
LCM	Life-Cycle Management
MANO	Management and Orchestration
MITM	Man-in-the-Middle
mMTC	Massive Machine-Type Communications
MNOS	Mimic Network Operating System
MSS	Maximum Segment Size
MTD	Moving Target Defense
MTND	Moving Target Network Defense
MUTE	Mutable Networks
NAC	Network Access Control
NASR	Network Address Space Randomization
NF	Network Function
NFV	Network Functions Virtualization
NFV-NS	Network Service
NGMN	Next Generation Mobile Networks Alliance
NIC	Network Interface Controller
NITRD	U.S. Networking and Information Technology Research and Development
NO	Network Operator
NSaaS	Network Slice as a Service
NSH	Network Service Header
NSM	Network Slice Manager
NSMF	Network Slice Management Function
NS	Network Slice
NTP	Network Time Protocol
OF-RHM	OpenFlow Random Host Mutation
ONF	Open Networking Foundation
OPEX	Operating Expense
OS	Operating System
OSS	Operations Support System
OTP	One-Time Password
P4	Programming Protocol-independent Packet Processors
PBR	Policy-Based Routing
PoC	Proof of Concept
PTP	Precision Time Protocol
QoS	Quality of Service
RDS	Reconnaissance Deception System
RRM	Random Route Mutation
RPAH	Random Port and Address Hopping
RTT	Round-Trip Time
SDN	Software-Defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
SMT	Satisfiability Modulo Theory
TPAH	TAP-based Port and Address Hopping
TOTP	Time-based One-Time Password
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
WAN	Wide Area Network

CHAPTER 1

Introduction

The societal impact of the Internet has increased over the years, gradually becoming a primary source for communication, information, entertainment, and socialization [1]. It is also a place to conduct business, being a backbone to sell products/services [2], and a tool for collaborative and remote working. Smart Cities, Smart Grids, Smart Crops, and newer forms of automation heavily rely on Massive Machine-Type Communications (mMTC), a scalability challenge for any network [3].

The revolution of Cloud Computing, supported by the advancements in virtualization, allowed for rapid elasticity, broad network access, on-demand self-service, and resource pooling [4]. Applications and Network Functions (NFs) alike can now scale according to the demand, repurposing free resources when demand is low and coping with seasonal spikes without over-provisioning resources upfront. All while allowing for a "pay as you go" model [5] or improved Operating Expense (OPEX)/Capital Expenditure (CAPEX) [6]. Consequently, there is a strong push to turn everything into cloud-enabled artifacts. Standardization efforts, such as the European Telecommunications Standards Institute (ETSI) Network Functions Virtualization (NFV) framework [7], introduced a Cloud Computing architecture suited for the telecoms' requirements. In particular, Virtual Network Functions (VNFs) have well-defined management components meant to integrate with existing Operations Support System (OSS)/Business Support System (BSS) and clear-purpose interfaces through which operations are carried out. Thanks to this, 5G, Internet of Things (IoT), Industry 4.0, and other new major networking trends are now cloud-enabled in some form [8][9][10].

Unfortunately, the increased popularity and business value also make the Cloud a desirable target to attack [11]. On top of that, the cloud computing model is encouraging a de-facto standardization of the same protocol stack (TCP/IP) [12] across previously segregated systems. For instance, the large numbers of sensors and actuators of IoT solutions are now becoming accessible over the TCP/IP protocol stack, as seen in Smart Grids, Smart Cities, or Smart Crops [13]. The IoT systems are being built on cloud-based infrastructures to cope with the many services and extensive data [14]. The use of standardized protocols to communicate with the cloud facilities, and the re-use of similar software solutions, substantially lowers the cost of an attack, as one vulnerability in a component of this stack now covers a broader range of potential targets.

However, we now have new controls available thanks to the softwarization of networks, as in Software-Defined Networking (SDN) [15]. The separation between control-plane and data-plane, along with SDN apps, allows addressing, in one go, functional features such as routing, simultaneously with security and dependability techniques, such as access control or multi-path [16]. In turn, virtualization allows

instantiating security middleboxes rapidly and on-demand. The capabilities given by softwarization and virtualization allow newer generations of networks to be more flexible and dynamic in dealing with the new threats posed by the cloudification of the world.

In this thesis, we have considered the threats posed by an attacker during the reconnaissance and delivery phases and introduced two novel dynamic security mechanisms to address them, leveraging the capabilities of softwarized and virtualized networks. Then, we optimized the use of the mechanism's most significant contribution in the context of network slicing, demonstrating varying resource usage and tolerance to changing network latency through dynamic reconfiguration.

The following sections will present the motivation and further details about the contributions.

1.1 Motivation

In today's typical computational systems, attackers have the upper hand against defenders, as time will favor their vulnerability scanning endeavors. Softwarized and virtualized networks are no exception. This asymmetric relationship between defenders and attackers motivates the introduction of the exploration space concept, which is meant to put an expiration date over the information acquired in those scans [17][18].

Moving Target Defense (MTD) is the research field that directly arose from the exploration space concept. Literature shows extensive research in all critical areas of MTD's taxonomy [18][19], with commercial offerings already appearing. By itself, MTD does not reduce the attack surface. The security improvements arise from the movement of that surface across the exploration space in ways unpredictable to the attacker, making the discovery and exploitation of a vulnerability harder.

However, there are practical issues with using MTD in larger-scale softwarized and virtualized networks (*e.g.,* 5G). Generating movement across clients/peers without disrupting legitimate usage, hindering other ancillary systems (such as auditing, accounting, and billing), or being detrimental to an attack's attribution and forensics [20] are some of the crucial issues. Synchronizing the movement across peers without incurring heavy overheads (synchronization protocol), sacrificing mutation speed, or connection reliability (*i.e.,* errors due to desynchronization) is still a significant challenge [21]. Furthermore, these networks require sufficient flexibility to add/remove peers at any time without disrupting the service to others.

This thesis' primary motivation is to deliver a practical MTD mechanism capable of solving today's deployment challenges and explore its capabilities to enhance defense further. Crucially, how to leverage the MTD approach to enhance the detection of malicious activity and further the incident response capabilities (*e.g.,* enable honeynets). Furthermore, it is essential to devise optimization strategies that alleviate the resources overhead required to run such an MTD system. Lastly, the conjunction of the devised solutions should be critically evaluated within a realistic virtualized and softwarized networks scenario.

The following subsections introduce the guiding research questions formulated during the thesis proposal, which this thesis has answered.

1.1.1 Research Questions

From the motivation and the initial thesis proposal, we get clearly defined research questions, formulated as follows:

Q1. How to dynamically change security in a softwarized and virtualized infrastructure without disrupting existing services or functionality? This question is the thesis' starting point:

how to leverage softwarized and virtualized networks' capabilities towards enabling dynamic security mechanisms. The mechanisms described in Chapters 4, 5 successfully answer **Q1** in greater detail.

Q2. Can concurrent defenses deployed dynamically and simultaneously improve security over a single defense? The most significant challenge underlying this question is whether the dynamic orchestration mechanisms would effectively deploy and configure multiple mechanisms to extract greater utility than a single defense with a much simpler configuration. The two mechanisms devised in this thesis were constructed having **Q2** in mind and successfully showed how two dynamically deployed defenses can improve security over a single defense. Nevertheless, a better formulation for Q2 would be *"Can an orchestrator extract greater utility from concurrent defenses deployed dynamically and simultaneously over a single non-orchestrated approach?"* The integration between the mechanisms described in Chapters 4, 5 successfully shows the answer is yes.

Q3. How does the effectiveness of concurrent defense mechanisms scale when related to the non-concurrent instantiations of each mechanism? Because the configuration space is limited, MTD mechanisms exploring an overlapping space could become counterproductive. The evaluation made while devising DynSec (Chapter 6) is straightforward in showing that the available configuration space is the only limitation. The practical aspects regarding orchestration are already addressed in Q2.

Q4. How would distributed locations, multi-tenancy, multi-ownership, and usage billing impact our dynamic security mechanisms? The most challenging question and fundamental for real-world applicability were answered in Chapter 7, with the integration into the H2020 5Growth. The interdomain communications were secured against probing and exploit delivery using the thesis' first MTD mechanism. The significant challenges with orchestration across the domains meant that multi-tenancy and multi-ownership were solved using a virtual slice extension, effectively creating a leveled playing field within that slice (*i.e.,* the owners and tenants are akin to peers). The H2020 5Growth Vertical Slicer addressed distributed locations using its built-in capabilities.

These questions are answered in greater detail in Chapters 4, 5, 6, and 7.

1.1.2 Research Goals

In particular, we have achieved the proposal's goals, which aimed to:

G1. Devise methods to change security parameters/requirements on-the-fly, without fundamentally changing the running functions. In effect, it allows dynamic instantiation and composability with existing network functions. Create mechanisms with configuration interfaces that leverage the orchestrators from virtualized networks, thus allowing changing security parameters on-the-fly.

G2. Research how the defense mechanisms, without any adversarial action, behave regarding service or functionality disruption. Because MTD relies on movement, legitimate interactions may fail due to missing calculations on where the target should be next. It is fundamental to determine if these false positives break the system's functionality or need further processing to eliminate the false positives.

G3. Investigate how such methods can be used to achieve an optimal balance between security, functionality, performance, and cost. This goal is part of the optimization efforts to reduce the computational cost of the MTD mechanisms while still maintaining sufficient movement to the modeled attacker.

G4. Establish ways that effectively coordinate defense across different locations, devices, and owners. That is, explore the orchestration capabilities and bridge the contributions

with the needs of real-world deployments.

G5. Critically evaluate the security benefit of defense mechanisms versus the cost and performance penalties incurred. This goal continues G3 and exists to establish a different milestone with a more evolved status within the performance optimization task.

G6. Retain the defender's home advantage while making the attacker fight a new battle at each new attempt. Crucial to make the proposed mechanism applicable to the real world and enable enhanced information gathering about the attacker. Furthermore, should the system collapse and the attacker be successful, this goal is fundamental to allowing a *post-mortem* analysis.

As a result, in the following section, we will present the achieved contributions of this thesis, also stating the earlier contributions that supported the thesis proposal.

1.2 Contribution

We have devised two MTD mechanisms in this thesis (seen in Chapters 4 and 5) that contributed to state-of-the-art. The first mechanism is a seamless Two Factor Authentication (2FA) MTD that uses Time-based One-Time Password (TOTP) and the service's ports to send authentication codes. This new approach avoids a dedicated synchronization protocol and does not introduce data overheads to perform the mutation. The second allows an active TCP connection handover across endpoints, thus enabling more sophisticated intelligence gathering (*e.g.,* honeynets). This approach complements the detection abilities of the first mechanism with seamless connection steering capabilities for the response.

The security provided by the mechanisms comes at the cost of computational overheads. As resources are finite, the defenses will only be workable if their overhead does not facilitate resource exhaustion attacks and functional requirements such as time constraints are still upheld. To do so, we have integrated our mechanisms with Management and Orchestration (MANO) orchestrators and experimented with their deployment. Furthermore, we have established a straightforward optimization method called DynSec that leverages the latency differences across the separate networks that allow network slicing and 5G network access. Our work contributes to the state-of-the-art by extending *Connel et al.* [22][23] research with multi-tenant and multi-owner scenarios with some degree of regular interference expected due to the shared access media, distribution, core networks, and regular resources. Consequently, the heterogeneity and velocity at which reconfiguration and requirements change will be significantly different than in a single-tenant, single-owner infrastructure with "homogeneous" resources, as in the prior art.

We have integrated the thesis mechanism within the pilots of H2020 5Growth and critically evaluated their feasibility, utility, and impact on the verticals using that stack. The results (Chapter 7) show the mechanism has a negligible impact on the functionality, providing a noticeable degree of protection against vulnerabilities exploitation, and has manageable overheads.

From a scientific point of view, the thesis work plan was designed and then iteratively improved (as per the scientific method) with the following expected results: **R1. Achieve the ability to change the security level of an NF dynamically.** This result sets the bar for the orchestration requirements and forces solving the challenges that would hinder adopting the proposed mechanisms in the real world. **R2. Improve the defense's cost-effectiveness.** Cost is a significant sore point in the real world and an excellent optimization parameter to be considered in the orchestration solution. **R3. Leverage the MTD target misses in more meaningful ways.** Go beyond the immediate defense capabilities created by the movement to disrupt the connection with the protected

function(s). Every time an attacker misses the target, there is a detection opportunity that incident response mechanisms could leverage. **R4. Deliver the right kind of security at every moment.** The dynamic orchestration and articulation are explicit with the different mechanisms and on-the-fly configuration parameters. **R5. Publish the results of each major milestone (seven in total) in an international conference, journal, or magazine.** Ensures the work is peer-reviewed and that the contributions are significant enough to be considered outcomes of this thesis. The goals were set before COVID-19 happened, which momentarily created severe publication challenges that had to be overcome. **R6. Contribute to standardization efforts that promote the furthering of security and privacy.** Perhaps a bit too ambitious during the proposal, the goal was to explore all avenues to further security and privacy standardization. **R7. Provide open challenges that allow master's students to conduct independent research to complete their degree requirements.** Identifying open issues and future work is essential to allow the onboarding of more manpower that may continue the research conducted in this thesis.

These results were achieved and demonstrated in Chapters 4, 5, 6, and 7. In the case of R3, the initial plan to onboard a Cyber Mimic Defense (CMD) mechanism proved to be hard to integrate and evaluate properly. However, once the in-depth evaluation of the MTD started, newer opportunities were presented that allowed to create new scientific questions geared towards incident response (and replace the original R3). We highlight this thesis's most significant scientific achievements in the following subsections, particularly in publications and other communication activities.

The thesis's primary outcomes in the publications and dissemination task consist mainly of international journal articles and international conference articles.

Journal articles with mechanisms of the thesis:

1. X. Li, C. Guimarães, G. Landi, J. Brenes, J. Mangues-Bafalluy, J. Baranda, D. Corujo, **V. A. Cunha**, J. Fonseca, J. Alegria, A. Orive, J. Ordonez-Lucena, P. Iovanna, C. Bernardos, A. Mourad, and X. Costa-Perez, "Multi-Domain Solutions for the Deployment of Private 5G Networks," IEEE Access, 2021.
2. **V. A. Cunha**, D. Corujo, J. P. Barraca, and R. L. Aguiar, "TOTP Moving Target Defense for sensitive network services," Pervasive and Mobile Computing, 2021.
3. **V. A. Cunha**, D. Corujo, J. P. Barraca, and R. L. Aguiar, "Moving Target Defense to set Network Slicing Security as a KPI," Internet Technology Letters, 2020.

International conference articles with mechanisms of the thesis:

1. **V. A. Cunha**, N. Maroulis, C. Papagianni, J. Sacido, M. Jimenez, F. Ubaldi, M. Gharbaoui, C. Chang, N. Koursioumpas, K. Tomakh, D. Corujo, J. P. Barraca, S. Barmpounakis, D. Kucherenko, A. Giorgetti, A. Boddi, L. Valcarenghi, O. Kolodiazhnyi, A. Zabala, J. Salvat, and A. Garcia-Saavedra, "5 Growth: Secure and Reliable Network Slicing for Verticals," 2021 European Conference on Networks and Communications (EuCNC), IEEE, 2021.
2. **V. A. Cunha**, D. Corujo, J. P. Barraca, and R. L. Aguiar, "Using Linux TCP connection repair for mid-session endpoint handover: a security enhancement use-case," IEEE SDN/NFV - Mobislice Workshop, IEEE, 2020.

Prior contributions that motivated the thesis proposal and early works:

1. **V. A. Cunha**, E. Silva, M. Carvalho, D. Corujo, J. P. Barraca, D. Gomes, L. Z. Granville, and R. L. Aguiar, "Network Slicing Security: Challenges and Directions," Internet Technology Letters, 2019.

2. **V. A. Cunha**, E. Silva, M. Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. P. D. Santos, L. Z. Granville, and R. L. Aguiar, "A Network Service for Preventing Data Leakage from IoT Cloud-assisted Equipment," in 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, Jun. 2019.
3. E. Sousa, **V. A. Cunha**, M. Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. P. D. Santos, L. Z. Granville, and R. L. Aguiar, "Orchestrating an SFC-enabled SSL/TLS traffic processing architecture using MANO," in 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, Nov. 2018.
4. **V. A. Cunha**, M. Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. P. D. Santos, L. Z. Granville, and R. L. Aguiar, "An SFC-enabled Approach for Processing SSL/TLS Encrypted Traffic in Future Enterprise Networks," in 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, Jun. 2018.
5. C. Parada, F. Fontes, C. Marques, **V. A. Cunha**, and C. Leitao, "Multi-Access Edge Computing: A 5G Technology," in 2018 European Conference on Networks and Communications (EuCNC). IEEE, 2018, pp. 277–9.
6. L. Marcuzzo, V. Garcia, **V. A. Cunha**, D. Corujo, J. P. Barraca, R. Aguiar, A. Schaeffer-Filho, L. Z. Granville, C. R. P. Santos, "Click-on-OSv: A Platform for Running Click-based Middleboxes," in 2017 IFIP/IEEE International Symp. on Integrated Network Management (IM), Lisbon, Portugal, May. 2017.
7. **V. A. Cunha**, I. D. Cardoso, J. P. Barraca, R. Aguiar, "Policy-Driven vCPE Through Dynamic Network Service Function Chaining," in 2016 IEEE International Conference on Network Softwarization (NetSoft), Seul, South Korea, Jun. 2016.

The primary MTD mechanism devised in this thesis (see Chapter 5) was featured as a demo in the EuCNC 2021 booth at the H2020 5Growth European project. Our booth won the award for the best booth of the conference. The MTD mechanism was also demoed in the EuCNC 2021 FAST workshop and received positive feedback from the attendees.

The thesis outcomes were communicated throughout the years in the Aveiro University's Research Summit. In the 2021 edition, the thesis won the best pitch of our session.

We have also participated in the early discussions of the IEEE Project 1912 (P1912)[1] Work Group and coauthored an early stage RFI to the NIST Privacy Framework[2] with the workgroup.

The research and results achieved within MTD and the incident response enabling mechanism contributed to the successful outcomes of Ana Gameiro's masters dissertation and Pedro Escaleira's ongoing master's dissertation efforts.

1.3 Thesis Structure

The thesis is structured as follows: in Chapter 2 the works related to the dynamic security mechanisms in softwarized and virtualized networks will be discussed, along with the standardization efforts and related works and trends that influence the capabilities of current softwarized and virtualized networks. Chapter 3 presents the related works, separated by field and relevance to this proposal. Chapter 4 introduces the thesis's first contribution, a workable MTD approach. Chapter 5 leverages the MTD mechanism's detection capabilities to introduce a smooth handover mechanism that enables enhanced response to an ongoing attack and performing information gathering. Chapter 6 introduces

[1] `https://standards.ieee.org/project/1912.html`
[2] `https://www.nist.gov/document/ieee1912theworkinggroupforieeeproject1912508pdf`

a straightforward metric (Key Performance Indicator (KPI)) to aid business verticals in optimizing MTD's resource usage and defensive capabilities. Chapter 7 documents this thesis dynamic security mechanisms integration efforts into real-world use cases, such as the 5Growth pilots, and discusses their contribution to those pilots. Finally, Chapter 8 summarizes this thesis's primary findings and contributions, critically discussing the achievements in light of the proposal and the integration made with real-world use-cases. The thesis concludes with future work and other enhancements that were not possible within this thesis time plan.

CHAPTER 2

Background

This chapter introduces and discusses the background works related to cybersecurity and the dynamic security mechanisms most relevant for softwarized and virtualized networks.

2.1 Cybersecurity

To introduce the dynamic security aspects of this proposal, we must first define what security means. In this context, security means adequately addressing the threat model of our assets without violating the cardinal rule (*i.e.,* never spend more than the asset is worth). Consequently, building the threat model and setting its delimitations requires understanding the different attack domains, expected attackers, and mechanisms used. To do so, we will start by introducing the fundamental security concepts and the typification of an attacker. We will then review MITRE's Common Attack Pattern Enumeration and Classification (CAPEC)[1], an evolving survey of the attack domains and adverserial mechanisms used in cyberspace. Afterward, we will describe the broad types of assets that may be targeted in the scope of this proposal. We will then model the attack process against our assets, using the Cyber Kill Chain [24] (and variants), completing the offensive side of the threat model.

At this point, we will need to start addressing the threat model to achieve security. We will identify what risk, exposure, and a threat is. Then, we will end this section characterizing the attack surface, briefly approaching the options to reduce this surface, and then defend it.

2.1.1 Fundamental Concepts

Information security is founded atop three fundamental concepts, often referred to as the CIA triad [25]: confidentiality, integrity, and availability. **Confidentiality** is the property in which the information, or its intrinsic characteristics, are not disclosed to unauthorized parties. In the context of data, **integrity** means maintaining the information's accuracy and completeness. In the context of systems, integrity means remaining within the expected logic and parameters, *i.e.,* the processes cannot be subverted by an unauthorized party. **Availability** is the property denoting that the authorized parties have access and can act upon the information (or system) as intended.

However, while the CIA triad lays the ground for the universal concepts of cybersecurity, these are also insufficient to describe security in some scenarios (such as softwarized and virtualized networks).

[1] `https://capec.mitre.org/`

As noted in *ISO/IEC 27000:2018* [26], more specialized properties such as authenticity, accountability, non-repudiation, and reliability may also be considered beyond the CIA triad. **Authenticity** means accurately identifying the source of information or the acting elements within a system. **Accountability** means accurately tracking resource usage or attributing actions to the actor that performed them. **Non-repudiation** means removing the ability to deny a fact or recognition of the obligations within a contract. **Reliability** denotes the property in which the desired operations are performed within expected parameters.

The extended fundamental security concepts will be used throughout this thesis, as defined in *ISO/IEC 27000:2018* [26].

2.1.2 The Attacker

The attacker is the threat actor against the system, with the intent of subverting at least one of the key security concepts. A system may be targeted by more than one attacker at a time, with different motivations and sophistication. An integral part of achieving information security is creating the threat model that defines who the attacker types against which the system must be defended are. It is unlikely that a security mechanism will be infallible or come without any cost. Therefore, identifying the attacker's anatomy is critical to understanding if an intended security solution is effective (or if a particular kind of attack is feasible). Furthermore, it also sets the delimitations of the security model. For instance, if the system is not required to survive attacks from state-sponsored actors (cyberwarriors), it is unreasonable to expect the security model to defend against those attacks explicitly.

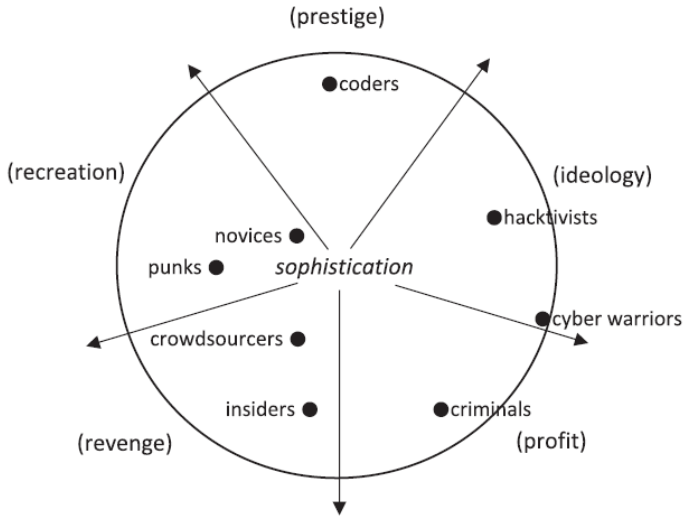


Figure 2.1: Circular order circumplex of hacker types [27]

An approach to understanding the attacker is using sociology works, such as *Seebruck's* [27] weighted arc circumplex model (Figure 2.1) that relates each hacker type and technical skill (*i.e.,* sophistication) with their motivations. In *Seebruck's* work, the motivations considered were recreation, prestige, ideology, profit, and revenge. Recreation refers to hackers driven simply by the technical challenge and the quest for offensive-security knowledge. Recreative hackers usually do not intend to cause harm, but they may cause collateral damage due to their exploratory actions. In turn, hackers that seek prestige want recognition (*e.g.,* social media attention), therefore, may cause some harm to prove their skills. Prestige hackers tend to measure success by the amount of recognition, not the scale of

actual damage inflicted to the running systems. Ideology, profit, and revenge are motivations driven by outside factors. The impact on the system will be determined by the opportunity and the objectives set by those outside factors. *Seebruck's* work helps identify the attacker types that may be driven by ideology, profit, and revenge. Thus, the threat models may assess those risks more accurately and devise adequate counter-measures.

After careful review, *Seebruck* categorized the attackers under the following hacker types: novices, punks, crowdsourcers, insiders, coders, criminals, hacktivists, and cyberwarriors. Novices are still exploring the trade and gaining knowledge of the attack process. Punks are mostly recreational attackers that are more sophisticated than novices. Crowdsourcers and insiders fall mainly on the revenge spectrum. Coders are professionals that seek recognition. Criminals, hacktivists, and cyberwarriors are the most sophisticated and relentless types of attackers. The circular order circumplex models (Figure 2.1) focus on the dominant motivation, showing a unidimensional representation of the motivation and sophistication of each hacker type.

Seebruck argues that in reality, hackers can be driven by more than just one motivation, therefore requiring a multidimensional representation of the motivation and sophistication of each hacker type (Figure 2.2)

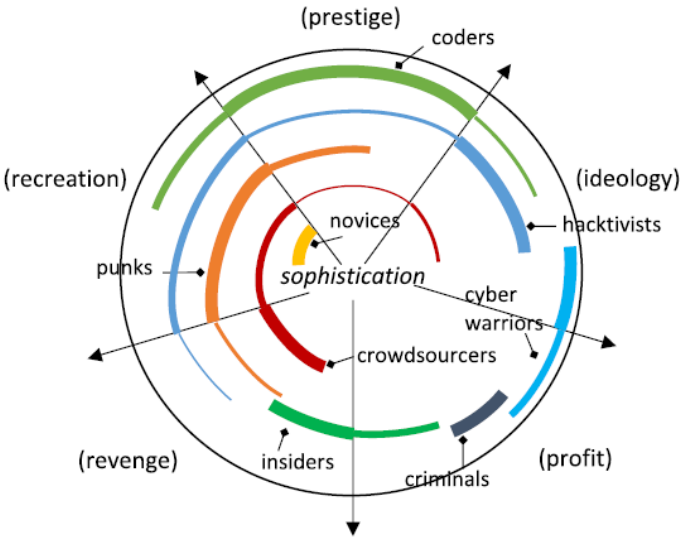


Figure 2.2: Weighted arc circumplex of hacker types [27]

Using the most well-known hacker teams as a case study, each threat actor falls into the following weighted arc circumplex (Figure 2.3). *Seebruck's* [27] work builds the bridge between the sociological aspects that drives the attack (*e.g.,* motivations and hacker types) and the sophistication required to carry them out. In particular, we can now look into the techniques used by the well-known hacker teams in this case study and compile the known/expected attack patterns to assess and defend against the motivations in a changing threat landscape. It is now possible to identify the attack domains and mechanisms used by those teams to create a better threat model and therefore develop more suitable defenses against hacker types with similar behavior.

The following subsections will introduce and describe the attack domains and mechanisms using MITRE's recommendations. These will be important in the later stages of the thesis to understand the effectiveness and delimitations set by the solutions.

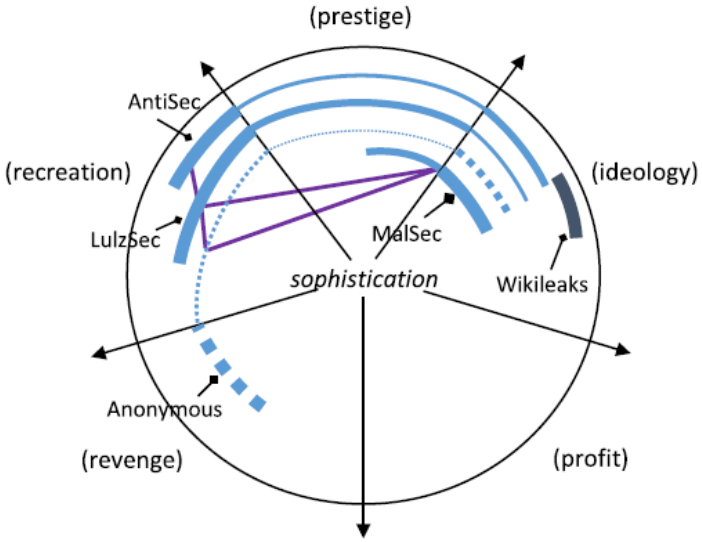


Figure 2.3: Weighted arc circumplex of hacker teams [27]

2.1.3 Attack Domains

The attack domains are the highest abstraction of the realms an attacker can operate and the threat model's starting point. In conventional terms, these would be the five dimensions of warfare: land, sea, air, space, and information. Determining the affected domains helps understand the types of personnel required to build an effective security solution. MITRE's ATT&CK Matrix for the Enterprise[2] gives a broad overview of an attack's anatomy against an Enterprise, similar to a kill chain. While MITRE's ATT&CK allows zooming into the details about each of the considered stages, the application focused MITRE's CAPEC-3000[3] already allows identifying the six attack domains clearly: communications, software, hardware, supply chain, social engineering, and physical security.

Communications or just Computer Networking encompasses the protocols, messages, and media used for computer communication. The techniques used here may block, modify, or steal messages to produce a negative outcome.

The **software** focuses on the exploitation of computer programs, from applications to the Operating System itself. The techniques in each attack pattern exploit weaknesses in the design or implementation of those programs, achieving a given negative impact. Unsurprisingly, the software is a big part of softwarized and virtualized networks.

Hardware denotes the direct exploitation of the physical side of the computing system, *i.e.,* the components that make the computations. The techniques used in this domain include the replacement, destruction, modification, and exploitation of design/implementation flaws of the hardware that makes up the system. The hardware is also a significant factor in softwarized and virtualized networks, particularly side-channel research.

Supply Chain focuses on compromising our systems through indirect attacks conducted against the third parties we rely on to construct the system. The techniques include modifying a component during manufacture or distribution (*e.g.,* adding a backdoor, a kill-switch, or another non-trivial fragility). With the cloudification push of softwarized and virtualized networks, the supply chain plays a more significant role in our system.

[2] `https://attack.mitre.org/`
[3] `https://capec.mitre.org/data/definitions/3000.html`

Social Engineering exploits the human characteristics of the system's user. The techniques here trick the users into leaking data or performing unwanted actions in the system, using similar deception tactics as fraud in the physical world. Social engineering is a factor to consider when developing the user interfaces that control the NFs and networks and determining the type and amount of information necessary so that the support staff makes the right decisions.

Physical Security denotes the non-computational side of physical elements that support the computing system, *e.g.,* the building where the system is placed. The techniques in this domain are closely related to common criminality in the physical world, such as physical theft, evading physical barriers, or physically disturbing the operational requirements of the computational system (such as cutting power or increasing the temperature). Physical security is still relevant when operating our infrastructure and an important consideration when choosing the supply chain.

Despite all of the identified attack domains being relevant to the formulation of a threat model in softwarized and virtualized networks, this does not mean that all domains must be present in all threat models at all times. Narrowing down the expected attackers allows delimitating attack domains, which helps deliver more effective security against the system's attackers. The following subsection will introduce the attack mechanisms.

2.1.4 Attack Mechanisms

The attack mechanisms are a lower form of abstraction of the attack patterns and the enablers of each stage described in MITRE's ATT&CK. The mechanisms are closely related to the type of actions taken over the attack domains, further detailing the attacker's behaviors, which must be thwarted. MITRE's CAPEC-1000[4] classifies the attack mechanisms into nine main categories, which are: engage in deceptive interactions, abuse of existing functionality, manipulate data structures, manipulate system resources, inject unexpected items, employ probabilistic techniques, manipulate timing and state, collect and analyze information, and subvert access control.

Engage in Deceptive Interactions, often known as "spoofing attacks," is an attack pattern in which the target is made to believe it is interacting with another principal. The objective is to usurp the level of trust between the target and the principal, unlocking privileges otherwise unavailable.

Abuse Existing Functionality is an attack pattern in which the adversary uses or manipulates existing application functions to overcome the expected application logic. This attack pattern may allow, for instance, disclosing unintended data, executing code, degrading performance, or denying service to others.

Manipulate Data Structures and their characteristics within the system, such as influence over the execution flow, to gain illegitimate access to data, privileges, code execution capabilities, or disrupt the service to others.

Manipulate System Resources is a broad pattern in which the attacker targets system resources (*e.g.,* computational infrastructure, files, libraries, or even configuration) to achieve his desired outcome. The system behavior is affected by changing resource states or availability, disrupting service availability, corrupting or disclosing information, executing arbitrary code, or bypassing system permissions.

Inject Unexpected Items is an attack pattern in which the attacker uses the input interfaces in unforeseen ways, hijacking the execution flow away from the intended behavior or the existing functionality. This method may lead to arbitrary code execution, break data integrity, disrupt service, or cause faulty behavior.

[4] `https://capec.mitre.org/data/definitions/1000.html`

Employ Probabilistic Techniques to infer knowledge over the behavior and assumptions of the system (*e.g.,* parameter fuzzing) or bypass known-secure protections that rely on a well-defined secret by using the probabilistic collisions within its space (*e.g.,* popular passwords or rainbow tables).

Manipulate Timing and State is an attack pattern where the attacker exploits race conditions to change the expected outcomes, modifies user-accessible state data to subvert the current system's state, or leverages unsound application design to cause a deadlock in the system (*i.e.,* deny service to other users).

Collect and Analyze Information is an attack pattern where various kinds of information are gathered about the system, protocols, fingerprints, the organization, and its users. The attack may either cause a direct disclosure of the targeted data or be a way to acquire information to succeed in another kind of attack.

Subvert Access Control denotes an attack pattern where the attacker exploits a weakness, limitation, or wrong assumption in the authentication or authorization mechanisms. As a result, the attacker may impersonate, gain illegitimate privilege, or have illegitimate access.

All of the above attack mechanisms are in the scope of softwarized and virtualized networks, being the suitability determined by the targeted asset. The following subsection will introduce the assets that must be considered.

2.1.5 Assets or Targets

According to the withdrawn *ISO 13335-1:2004* [28], which is still widely used by normative bodies such as Enisa, an asset is anything of value to the organization. From the defenders' perspective, the assets are the elements of the system, which are valuable enough to warrant some kind of protection. The *cardinal rule* states that no asset warrants more protection than its value. From the attacker's perspective, the assets are the targets of an attack. In particular to our thesis, typical assets include information, software, service availability, intangibles such as corporate image, and the hardware that provides these services.

About the goals of the proposal, *Ward et al.* [19] surveyed a list of the entities being protected in systems that already employed some dynamic security mechanism. Those assets reflect the previously mentioned attack domains and attack mechanisms, being the assets: stored data, traffic, session, network, applications, operating system, and machine.

Stored Data needs to be protected against information disclosure, corruption, and data loss. Other forms of information (meta-data) must also be attended to, as it may provide ways to exploit the organization. **Traffic** is data plus control meta-data (such as the protocol stack) being transported through a network. On top of information disclosure and corruption, malicious data may be injected into the network, compromising the system integrity or aiding in disrupting availability. A **session** is a context that frames a set of user interactions, maintaining the established trust between parties. The compromise of a session allows usurping trust. A **network** is the computer communications infrastructure that allows traffic to be sent across end-points. It must be protected against denial of service attacks, service degradation attacks, traffic interception, and spoofing attacks. **Applications** are computer programs that interact with users, network traffic, or stored data. They must be protected against attacks coming from the network and local attacks from bad users or malicious applications. A compromise of an application may result in information disclosure, data corruption, data loss, disruption of service availability, degraded service, privilege abuse, or malicious code execution. **Operating System** is the platform that allows running the application. It needs protection from remote network entities and malicious programs running in the same system, especially privilege escalation exploits,

access to kernel-space memory, or illegitimate access to another application's memory. A **machine** is the aggregate of computational resources, virtual or otherwise, that run a given system. A compromise at this level has a severe impact, allowing for all of the above nefarious actions.

Notwithstanding, although focused on the proposal, the above assets are just for reference. There must be an asset determination on a case-by-case basis.

2.1.6 Modeling the Attack Process

Having already presented the attack domains and mechanisms used by the attackers to abuse a given asset, the process through which the attacker ties together its actions during an attack remains to be introduced. Understanding the attack process will help create more effective barriers, prioritizing the defensive actions which matter the most to achieve security within the threat model.

Hutchins et al. [24] proposed a "Kill Chain" to model computer network attacks or computer network espionage. The proposal later became known as Lockheed-Martin's "Cyber Kill Chain (CKC)," which became the *de facto* standard to model the network attack process in cybersecurity. The CKC (Figure 2.4) comprises seven stages: reconnaissance, weaponization, delivery, exploitation, installation, Command and Control (C2), and actions on the objectives.

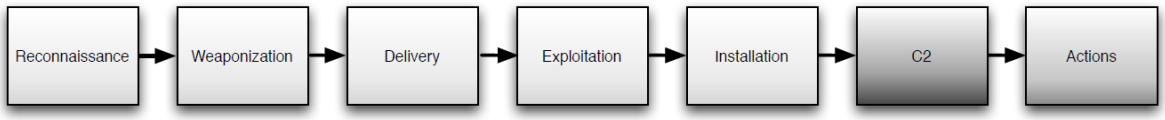
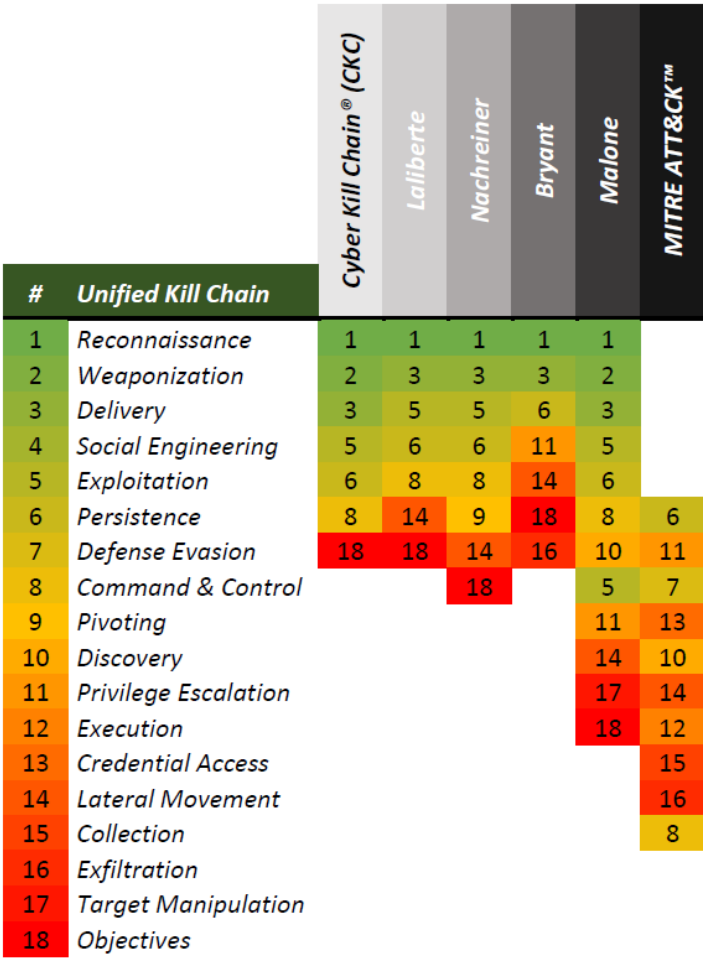


Figure 2.4: Lockheed-Martin's "Cyber Kill Chain (CKC)" [24]

Reconnaissance is when information about the organization is gathered, the services and systems being exposed, who the organization members are, whom they interact with, and how. The information gathered contains data about the organization's activities, personal data about the people within the organization and outside personnel that interact with it (and how), and technical details of the services and systems being exposed. **Weaponization** follows, identifying the vulnerabilities from all the information gathered in the previous stage and building the deliverables (as known as payloads) that exploit those vulnerabilities. **Delivery** is the phase in which payloads will reach the target systems, working around the difficulties of not having a foothold on the target systems, ensuring the payloads are put into place for later execution. **Exploitation** is when the payloads are executed, and a precarious foothold is achieved within the target system. **Installation** is a critical stage in which the attacker needs to gather as much information as possible about the compromised system while being the most exposed to detection. The acquired information is used to devise a way to consolidate the foothold. **C2** is the phase in which persisting access to the compromised system is established, creating a reliable communications channel to act upon the system. **Actions on Objectives** is the final stage, in which the attacker can now use the compromised systems to achieve its goals.

However, the CKC is not without some criticism. Academia and industry alike point out CKC's lack of attack modeling after breaching the perimeter, while some authors argue that stages unactionable by the defender could be removed or replaced. Because of this, some authors proposed variants of the CKC, while others presented new models. *Pols et al.* [29] comprehensively surveyed and compared the different kill chains that arose after the CKC, using prominent attacks from known hack groups as a case study, ultimately proposing the "Unified Kill Chain (UFK)." The comparison (shown in Figure 2.5) is made using the 18 stages that end up making the UFK, which are then related to the stages of each kill chain. This comparison considered five kill chains, of which four are modified

CKCs, while the last is a new model. *Laliberte*[5] modified the CKC to have Javascript malware in mind, replacing the unactionable weaponization stage for a lateral movement one. *Nachreiner*'s[6] Kill Chain 3.0 also removed the unactionable weaponization stage, similarly highlighting the need for lateral movement. *Bryant et al.* [30] framework modified the CKC to facilitate logical alarm data aggregation in a relational database. *Malone* [31] argued that the CKC only addresses the perimeter, therefore, extended the CKC to address the internal and target manipulation parts. The previously mentioned MITRE's ATT&CK[7] model focuses on the actions taken after breaching the perimeter.



#	Unified Kill Chain	Cyber Kill Chain® (CKC)	Laliberte	Nachreiner	Bryant	Malone	MITRE ATT&CK™
1	Reconnaissance	1	1	1	1	1	
2	Weaponization	2	3	3	3	2	
3	Delivery	3	5	5	6	3	
4	Social Engineering	5	6	6	11	5	
5	Exploitation	6	8	8	14	6	
6	Persistence	8	14	9	18	8	6
7	Defense Evasion	18	18	14	16	10	11
8	Command & Control			18		5	7
9	Pivoting					11	13
10	Discovery					14	10
11	Privilege Escalation					17	14
12	Execution					18	12
13	Credential Access						15
14	Lateral Movement						16
15	Collection						8
16	Exfiltration						
17	Target Manipulation						
18	Objectives						

Figure 2.5: The Unified Kill Chain and its stages compared to the other kill chains [29]

For this thesis, the attack process considered will be adjusted according to the assets being protected. If the protection is better characterized as perimeter defense, then the CKC will likely be preferred, as it is a *de-facto* standard. In turn, when defense-in-depth is required, one of the latter options will likely be chosen (UFK, *Malone* [31], or MITRE's ATT&CK). Selecting the proper attack process will allow defending more effectively, quicker, and at a lower effort. However, to best characterize the required defense and choose the proper process, we must first understand the attack surface of the asset.

[5] `http://www.darkreading.com/attacks-breaches/a-twist-on-the-cyber-kill-chain-defending-against-a-javascript-malware-att`
`a/d-id/1326952` *[Last accessed: 2021-10-29]*

[6] `https://www.helpnetsecurity.com/2015/02/10/kill-chain-30-update-the-cyber-kill-chain-for-better-defense/`
[Last accessed: 2021-10-29]

[7] `https://attack.mitre.org/`

2.1.7 Attack Surface

The attack surface of our assets and how to reduce and defend it are the last entries required to describe the more traditional cybersecurity defense approaches and, in the following section, introduce the dynamic defense approaches.

In order to define what is an attack surface, we must first introduce the correct terminology to define risk, exposure, vulnerability, and threat. According to *ISO 31000:2018* [32], the ***risk*** is the effect of uncertainty upon the objectives. That effect is the deviation from the expected outcome (regardless of whether that deviation has a positive or negative outcome). In our context, the standard states that risk can be regarded as (1) the likelihood of something happening or (2) the event's impact if it occurs. *MITRE's CVE Terminology*[8] defines ***exposure*** as a configuration issue or mistake in software that is not in itself the total compromise of the system but rather a stepping stone into fulfilling the compromise. When exploiting that issue leads by itself to a compromise of the system, we have a ***vulnerability***. A ***Threat*** is defined as the potential to cause an unwanted impact on a system or organization (*ISO 13335-1:2004* [28]). That is, using MITRE's terminology, threats are vulnerabilities and exposures.

Therefore, the threat model deals with vulnerabilities and exposures, managing risk according to the cardinal rule, the delimitations set when defining the security model, and other business factors. However, to manage anything, we must first have ways of measurement so that the actions can be pondered.

Manadhata et al. [33][34] introduced the attack surface concept to systematically measure the risk and threat of a computer system. The attack surface spans across the different attack domains, increasing with each applicable attack mechanism, even if currently unsuccessful in its exploitation. The authors concluded that in the software attack domain, the larger the number of lines of code, used components, or publicly available services, the higher the threat, thus the more extensive the attack surface. *OWASP*[9] reaches a similar conclusion, defining the attack surface as the number of entry-points available, the computer code which protects the entry-points, the data within the application, and the computer code which protects the application's data.

Reducing the attack surface in the software attack domain requires redesigning and redeveloping existing solutions to limit the number of entry-points, services, used components, and (in general) lines of code that control the application logic. Thus, this is an expensive and labor-intensive task. Making matters worse, because an application (or service) can only be useful if it has (at least) one entry-point of data or holds within it some valuable data, the attack surface can never be zero.

Holistic and systematic ways to reduce the attack surface across all attack domains also exist, such as complying with an organizational security standard like the *ISO 27001:2013* [35]. These standards will have (amongst others) guidelines to monitor vulnerabilities and processes to develop and deploy the corrections to said vulnerabilities across the various attack domains.

The attack surface is defended by introducing additional controls. In the software attack domain, those controls are functions such as firewalls, Intrusion Detection Systems (IDSs), Intrusion Protection Systems (IPSs), Anti-virus, application jails, containerization, or virtualization. Beware the controls themselves also contribute to the attack surface, possibly increasing it, as they will have exposure and vulnerabilities of their own.

Therefore, traditional defense strategies balance reducing the attack surface and introducing new

[8] `https://cve.mitre.org/about/terminology.html`

[9] `https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.md`

control functions. The makers of a given network service, hardware, or physical infrastructure, can weigh the cost of reducing the attack surface versus adapting the delimitations of their security model and issuing new defensive guidelines. Their customers can only defend the assets, change vendors, or use negotiation to influence the decision made by the makers.

2.2 Dynamic Security Mechanisms

The dynamic security mechanisms were introduced to address the practical shortcomings of the common cybersecurity defenses. These shortcomings will be detailed in the asymmetric relationship subsection, followed by the existing solutions presented in the exploration space subsection.

2.2.1 The Asymmetric Relationship

As the attack surface of any useful application will always be non-zero, we need to determine the balances between attackers and defenders to ascertain how a defense of that surface can remain effective.

Given how the attack process is modeled (Subsection 2.1.6), *Jajodia et al.* [36] asserts in their findings a long-term advantage in favor of the attacker's endeavors. The asymmetric nature of this relationship is attributed to three root causes: (1) the certainty in the network applications/services, (2) network topologies do not change in unpredictable ways, and (3) the homogeneity inside the network.

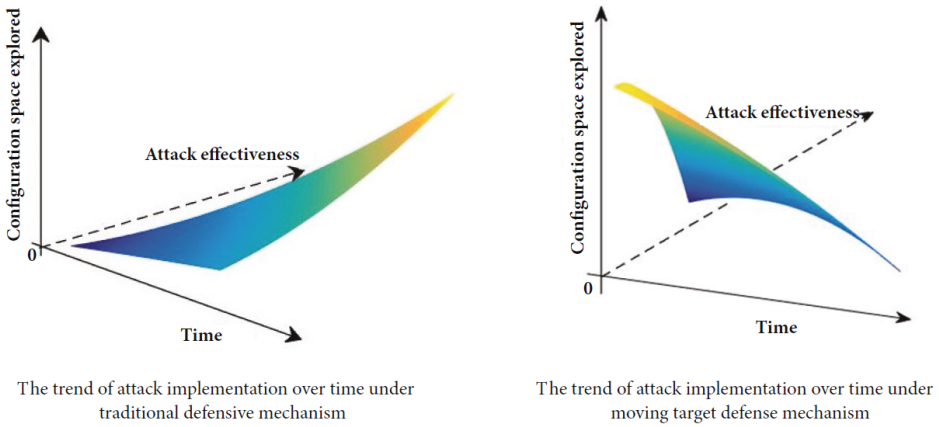
Network applications are meant to be accessible so that their legitimate users can easily find and use them. Thus, most applications forgo service discovery, opting instead for a stale and publicly available configuration which the users will always use to reach the service. This staleness aids the attackers in reaching the network functions that provide the service, even if they are illegitimate users. When this configuration is not public (*e.g.,* hidden inside a private client application), the attackers can still gather enough information to quickly find the systems providing access to the services by tapping into the network and analyzing the unprotected layers of the protocols. Then, taking as long as necessary to prevent triggering any defenses, the attackers can probe and discover more entry-points to the services and previously unknown vulnerabilities (zero-day). Thus, the attackers' knowledge and ability to attack the service improves over time. Unfortunately, the defenders' knowledge of unknown vulnerabilities will not improve until an attacker exploits or otherwise discloses the zero-day. Therefore, time is at the attackers' advantage.

Traditional network management paradigms favor predictability and human understanding of the topology, as (ultimately) it will be up to human administrators to diagnose and fix any misbehavior. Unfortunately, this predictability also favors the attackers. The longer the attacker can observe the network, the more and better information he will acquire to establish C2 or exfiltration channels. The human management approach also implies some administrative tasks that cannot be done in real-time, such as responding to equipment failure or security patching networking equipment. Therefore, the attackers can leverage the fact that the defenders will struggle to respond timely to some actions.

Most networks and terminals that connect to the network tend to be purchased in bulk and configured similarly. They usually have the same software installed to alleviate the management tasks required to keep these elements functioning. Similarly, this approach helps maintain a comprehensive defense strategy (*e.g.,* knowing which vulnerabilities exist in each terminal and deploying the security patches timely). The heterogeneity is not a feature by design but rather an unintended consequence caused, *e.g.,* by legacy and waves of purchases. Therefore, there is likely homogeneity within the devices of the same wave. However, sharing similar security traits across devices also includes sharing

the same exploitation vectors, lowering the cost of the attack to compromise a more significant number of devices. This fact highlights the cost advantage that the attackers have over the defenders.

Figure 2.6 illustrates the attack effectiveness over time. The plot on the left shows how the traditional cybersecurity defenses fare over time, while the one on the right shows the dynamic security defenses. Because traditional defenses tend to remain stationary in their configuration, as time goes by (purple to yellow), the attacker can acquire and correlate more information, which increases the ability to perform an effective attack. Incidentally, the attacker also can actively probe for vulnerabilities in the production system while avoiding detection or working around the defense systems. In contrast, the dynamic security defenses over time are like a wrap-around concave sail surface. Starting at instant zero (purple), the attacker can gather information and increase the attack effectiveness (like in the traditional defenses) until the system starts mutating its configurations. From there forward, we see the configuration space increasing, and the attacker starts losing actionable information. As time moves forward, the attacker will practically remain in the same effectiveness standing as instant zero (yellow). This configuration space is further explored in the following subsection.



The trend of attack implementation over time under traditional defensive mechanism

The trend of attack implementation over time under moving target defense mechanism

Figure 2.6: Comparing the attack effectiveness over time [17]

2.2.2 Exploration Space

The exploration space denotes all possible system configurations/mutations achievable in our network. The attackers will be probing this space during the information gathering phase in search of targets, exposure, and vulnerabilities. Our attack surface represents a particular configuration of our system; therefore, it is always within the exploration space. As shown in Figure 2.7, current systems are usually suited to the service they are running (most of the attack surface), therefore, have a relatively small exploration space. Traditional network defense reduces the attack surface as much as possible (*e.g.,* reducing the number of entry-points and patch bugs) or introduces new security controls to fortify the exposed surfaces (*e.g.,* access control, firewalls, or IDS/IPS).

Approaches such as MTD [37], [38] suggest that we should also increase the exploration space as much as possible and then move (in unpredictable ways) our attack surface over that space (as shown at the bottom of Figure 2.7). The rationale for this choice comes from the asymmetric relationship the attacker has over the defender (Subsection 2.2.1), in which the attacker has time as an advantage to achieve his goals. By periodically changing the configuration (in unpredictable ways), the information gathered by the attacker now has an expiration date, removing the time advantage of the attacker.

Due to the significance of MTD to this proposal, a longer historical context about its creation and the original motivations will be given so that a broader picture of the current state of the field and

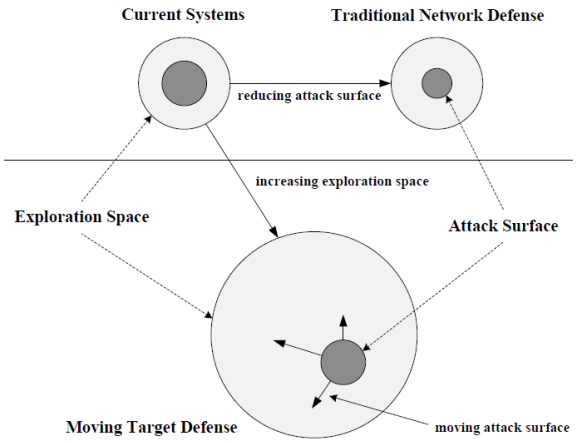


Figure 2.7: The Exploration Space [18]

future directions can be drawn. MTD is an active defense method that is inspired by target practicing at the shooting range. As in the "sitting duck" saying, empirical knowledge implies it is easier to hit targets when they are stationary than when they are moving. Because Network Services (NFV-NSs) behave like static targets most of the time, the idea behind MTD was that (somehow) moving around those services would increase their exploitation difficulty.

The first proposal was made during a summit organized by the U.S. Networking and Information Technology Research and Development (NITRD), called the "National Cyber Leap Year Summit 2009"[10], in which eleven raw ideas kick-started the field [37]:

1. **Mutable Networks: Frequently Randomized Changing of Network Addresses and Responses.** The idea is to continuously rotate the resolved addresses of VNFs so that each instantiation is exposed just for the shortest amount of time. Additional resilience would be achieved through the use of different Operating Systems. This approach is suitable for short interactions with the NF. The primary aim was to make scanning and reconnaissance more difficult.
2. **Diversity in Software.** Typically organizations create a software monoculture. That is, there is the tendency to install the same set of programs across all machines. Consequently, once an attacker finds a viable exploitation path in one machine, he will likely be able to exploit many other machines (if not all) following the same path. The core idea is that diversifying the installed software across the organization's machines would force the attacker to find different exploitation paths for each machine, making it harder to compromise large parts of the organization.
3. **Robust Random Authentication.** Essentially the way an individual authenticates to the system would vary randomly and according to the task or user's context. At that time, biometric authentication (such as fingerprint reader) or 2FA was not widely used in the general consumer world. Although some corporations had already adopted those authentication methods, the inherited cost of a not mass-market solution meant these solutions were not deployed unless the perceived threat justified the added cost. Today this cost is much lower; therefore, these solutions are more widely deployed.
4. **Resilient Cryptographic Systems.** Most cryptographic systems can be compromised by a single point of failures, such as a failure of the randomizer, an incorrect implementation, a compromise of the private key, a side-channel, a bug, an hardware failure, supply-chain attacks,

[10] `https://www.nitrd.gov/nitrdgroups/index.php?title=National_Cyber_Leap_Year_Summit_2009`

weak cryptography standards, loss of physical security, or novel attacks (such as quantum computing). The idea was to create new cryptographic systems requiring multiple failures before becoming compromised (through active checking, redundancy in calculations, or more resilient design).

5. **Connectivity Diversity.** It focuses on the network topology and how an NF is reached, addressing attacks like Denial of Service (DoS) or Man-in-the-Middle (MITM). The idea is that intelligent Sense-and-Respond mechanisms would allow for sufficient network path diversity to eliminate single points of failure, such as an attacker finding a good interception point or the weakest network link to saturate. Although the mindset and approaches predate SDN, apart from protocol choices (implementation), the ideas are still relevant for SDN-enabled networks.
6. **Decoys.** The participants state that any piece of software will invariably have a bug, which may later be exploited as a security vulnerability. Therefore, having that NF exploited is only a matter of time. By adding decoy NFs, one can delay the attacker's efforts. While purposefully increasing the attack surface, it diminishes the possibility of an actual breach of an NF that carries real data. Furthermore, the decoys can be used to study the attacker's methods and a deception tool to misdirect attackers from more valuable targets.
7. **Configuration-Space Randomization for Infrastructure.** The configuration determines how different NFs will be logically integrated into an end-to-end service. To do so, it holds valuable information about the relationships between the different NFs and how they communicate. This information allows the attacker to determine the higher value targets and the paths available to reach them. These paths can be either direct or through middle components, being the compromise of any middle component also aided by the knowledge of its configuration. Therefore, the idea is to create meta-configurations that describe the intended service logic, being the actual configuration deployed in the NFs a unique random instantiation of that service (*i.e.,* the NFs may be configured very differently at the network level, but still ultimately deliver the same service logic).
8. The **Distributed Data Shell Game.** The core idea is to break data into many pieces, distributing those pieces across separated systems (*e.g.,* BitTorrent). Therefore, when a catastrophic event happens, there will be additional resilience against data loss, as more systems will have to be disrupted to compromise the data.
9. **Security on Demand.** Proposes a mindset shift from designing systems that rely on "keep the bad guys away" to assuming from the get-go that applications will run on fundamentally insecure systems. Therefore, the question becomes how to temporarily establish sufficient security within that insecure system (against a network attack), as in a short-lived cocoon. Intel SGX[11] is a commercial implementation of this idea, which has flaws of its own (*e.g., CVE-2018-3615*).
10. **Chaotic[12] Organization Model.** The attendees suggested copying the decentralized nature of chaotic groups (and their cells). They claim those groups are hard to infiltrate, immune to significant losses if one faction is compromised, and they can work autonomously without needing an extensive rule set. This organization model was thought to be a game-changing idea, breaking away from linear control/command, tight communications, and high dependency of operations as in traditional processes. Loose ties, ample autonomy, and self-organized leadership are the appealing traits of the *Chaotic Organization Model*. However, it is unclear if such a model would even be viable. For starters, there are profound ethical implications in making public any workable methods that can benefit said organizations. Then, the "experts" of the

[11] `https://software.intel.com/en-us/sgx/details`
[12] *non verbatim*

field are notoriously uncooperative and elusive, making it harder to translate existing operational methods into computer science analogs. We must also consider the possibility that inflicting *any* significant damage to a common target may not be as demanding as collaboratively building an objective. One allows for many degrees of freedom and even rationalization after the fact, while the other has a minimal window to work right. As a final note, the author's intuition is that guerrilla warfare methods would be much more effective in MTD than chaotic organization methods.

11. **Smart Motion Adaptation Management.** *Smart Management* would holistically move across the infrastructure, both NFs and network connectivity, in an unpredictable way to the attacker. The motion models could be determined by game theory, machine learning, statistical analysis, or other dynamic control algorithms. This approach allows for control over the security-performance trade-offs, being the performance likely improved over other solutions.

The MTD concept was then formalized when NITRD released the "Cybersecurity Game-Change Research & Development Recommendations" [38]. There were some notable differences in the ideas from the summit. For instance, the cocoon in the *Security on Demand* proposal originated a new theme known as *Tailored Trustworthy Spaces*, which exists independently of MTD. The *Chaotic Organization Model* is no longer mentioned, neither are its goals directly incorporated into MTD. The field shifted towards a higher-level description of end-goals, rather than engineering topics, from the intuition of the expected implementation. These goals closely relate to the vision of creating, evaluating, and deploying dynamic security mechanisms which increase the system's reliability by reducing the vulnerabilities exposure and opportunities for attacks. The goal statement encourages:

1. Design resilient systems that can be reliable in compromised environments.
2. Move from a reactive security mindset to a more preemptive one.
3. Ensure the dynamic mechanisms only disrupt the attackers, not the end-users or service management.
4. Study the effectiveness of the dynamic mechanisms against different types of attacks.
5. Learn from the attackers as they try to break our systems, exposing their actions when needed.

The research that followed, which spanned over nearly a decade, comprises works that can be organized in taxonomy, as shown in Figure 2.8. This taxonomy closely follows the one presented in the survey conducted by *Zhou et al.* [18], which builds upon *Ward et al.* taxonomy of MTD techniques [19], further elaborating on the sub-field of Moving Target Network Defense (MTND) [18][39][40]. It is also consistent with a separate survey by *Lei et al.* [17] and the candidate's independent research.

MTD can be divided into five different subfields (Figure 2.8):

1. **Dynamic Networks** or **MTND** is a sub-field of MTD which focuses on the network connectivity side of applications. The subfield goes from the protocols, parameters, or configurations of the NF, to the topology, route, forwarding graph, and network equipment protocols and configurations, along with everything in-between. It is also the field under which most of the expected contributions of this thesis will fall.
2. **Dynamic Platform** changes the properties of computational resources, such as CPU architecture, OS version, platform data format, and others.
3. **Dynamic Runtime Environment** includes changes to the addressing space presented to the application or the instruction set accepted by the Operating System.
4. **Dynamic Software** changes the running code on the fly, modifying the instructions, order, grouping, or format.

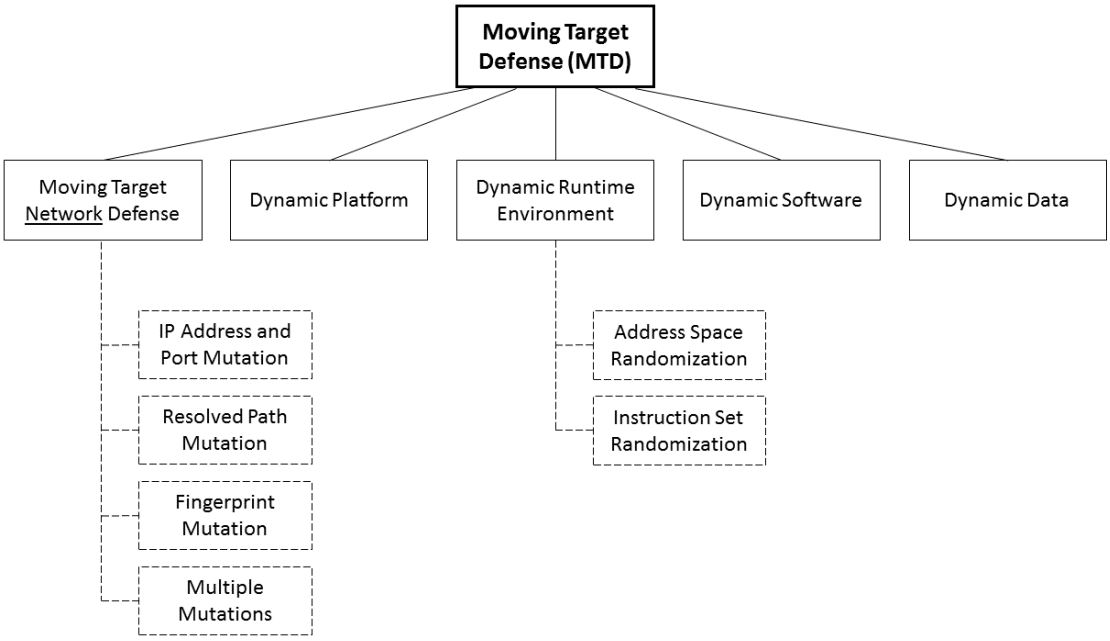


Figure 2.8: Taxonomy of existing works related to MTD [18][19]

5. **Dynamic Data** changes the encoding, format, syntax, or application data representation on the fly.

As softwarized and virtualized networks have a strong push towards cloud-based architectures, all of the above subfields may be relevant to developing this thesis's dynamic security mechanisms. Due to the characteristics of MTD, which moves the attack surface over the exploration space, the thesis dynamic security mechanisms may allow adjusting the security level on the fly (by limiting the amount of used exploration space or changing the time between mutations). It may also allow for all sorts of optimizations, from infrastructure resource utilization to improving business KPIs such as latency, all while assuring the right amount of security for that threat model (at that time). However, while MTD should make exploitation harder, it is not meant to patch the vulnerability. Therefore, there is a non-zero probability of an attacker successfully exploiting a vulnerability in a network service protected by MTD.

2.2.3 Related Economics Principles

As security is a balancing act of different choices, with varying costs and expected benefits over the system, some existing economic principles can aid decision-making. We will introduce the concept of Marginal Utility, followed by the Cost-Benefit Analysis (CBA), and the Cost-Effectiveness Analysis (CEA).

2.2.3.1 Marginal Utility Function

In economics, the *utility* represents the benefit taken from the consumption of a good (or resource). Therefore, the utility is quantifiable but subjective as a metric, as each subject may get a different benefit from the consumption of the same good. The *marginal utility* of a good (or resource) is the change in utility one gets from increasing/decreasing the consumption of that resource.

The *marginal utility function* is the mathematical equation that denotes, for that particular subject (or like-minded individuals), how the utility can be quantified for every possible change in consumption.

2.2.3.2 Cost-Benefit Analysis (CBA)

The CBA is an economics optimization process that assigns a value (*e.g.,* effort or directly monetary) to measure the effect (benefit). This process is a systematic approach to estimate the strengths and weaknesses of each choice, optimizing for the one that delivers the most benefit for a unit of cost.

2.2.3.3 Cost-Effectiveness Analysis (CEA)

The CEA (like Cost-Utility Analysis) does not attribute a monetary value to the measure of the effect (benefit), as that effect may be inappropriate to quantify in monetary terms (*e.g.,* health-care). Therefore, CEA is typically expressed as a ratio between the expected benefit (*e.g.,* extra years of life) and its cost. The optimization is geared towards maximizing the average level of the benefit.

2.2.3.4 FinOps

The related economic principles in this section already motivated the creation of FinOps[13], a financial operations and management cycle for cloud deployments. The cycle consists of three main phases (as show in Figure 2.9): inform, optimize and operate. The inform phase creates cost visibility so that teams can decide how to allocate their resources and share accountability. The optimize phase allows identifying the metrics and opportunities for resource optimization. Finally, the operate phase executes the processes and decisions carried in the previous two phases.

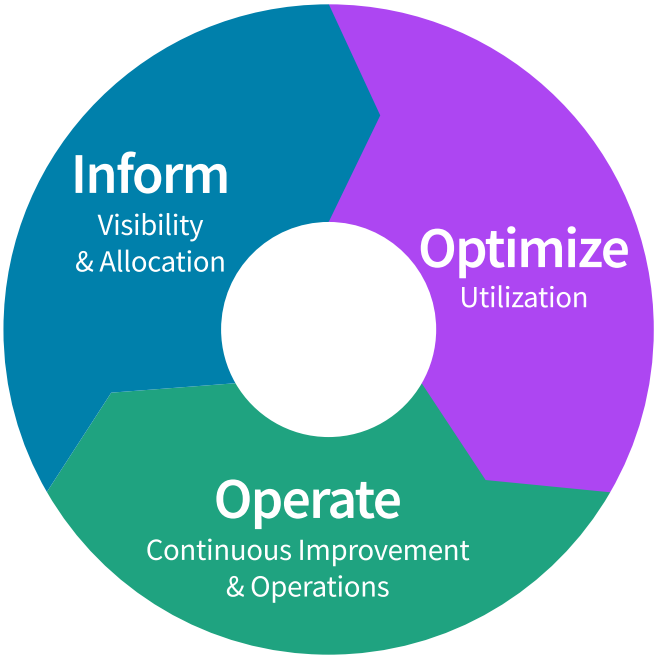


Figure 2.9: FinOps phases

[13] `https://www.finops.org/`

2.3 Summary

The background section highlighted the need for correct problem formulation in cybersecurity. In particular, the multi-disciplinary definition of the attack model and its delimitations will determine the issues that the cybersecurity solution must address and its suitability. The already identified security challenges of softwarized and virtualized networks (such as 5G) and the expected pervasiveness of this type of network set the motivation for this proposal. The dynamic security concept and underlying methods set the approach to solving the motivating challenges. Lastly, the brief economic principles present the optimization guidelines to bridge dynamic security capabilities with real-world management and optimization needs.

CHAPTER 3

Related Works

This chapter presents the related works, separated by field and relevance to this proposal. The chapter begins by relating some background in softwarized and virtualized networks with the surveyed security-related issues in those fields. Then, the works in MTD and CMD will be reviewed. Lastly, the closing remarks will bridge the existing works with the objectives set by this thesis motivation.

3.1 Softwarized and Virtualized Networks

This section will introduce the softwarized and virtualized networks, starting with their concepts, architectures, protocols, and features. Then, we will go further into the security aspects already identified in each entry.

3.1.1 Software-Defined Networking (SDN)

According to the Open Networking Foundation (ONF), there are four basic principles for an SDN architecture [15]: decoupling data from the control (*i.e.,* establishing the control-plane), logically centralized control, the programmability of network services, and open interfaces.

Decoupling data from the control allows deploying the entities that perform the control independently from forwarding and processing traffic. This decoupling effectively establishes a control plane separate from the forwarding plane. The control plane separation allows for specific platform optimization with different software life cycles. It also serves as a pre-condition to attain central control through an SDN controller, which has management capabilities over the resource groups. These resource groups are considered a part of the data plane, as they will be associated mainly with the processing of client-generated traffic. In particular, for this thesis, this decoupling translates to the enforcement of packet forwarding rules and the processing of data within network flows being separated from the decision mechanisms which determine where to forward and how to process that data. This design is beneficial to our proposal, as it reduces the ability of an external attacker to compromise our dynamic security solution.

Logically centralized control means that regardless of the distributed or monolithic nature of the implementation, the control will always act as a single entity that makes decisions over the network. This design is beneficial for this thesis' security purposes since the various enforcement elements do not have to determine which is their current controller with authority or the right enforcement action to

be taken, as only a single centralized control exists that always sends correct actions for enforcement. Additionally, the network behavior becomes easier to test and reproduce, as all control actions are determined by a single entity that performs discrete recordable actions over enforcement elements.

The **programmability of network services** allows exchanging information with the SDN controller, be it during service discovery, negotiation before establishing the service, or throughout the whole service lifecycle. This information exchange allows for providing intricate details about the client and service states, which are valuable to the decision processes of a dynamic security solution. In broader terms, ONF's goals were to disentangle the client's purpose from the resource operation, making clients less complicated and more independent of the network infrastructure. A tradeoff exists between client simplicity and participation in fine-tuning (or continuing optimization), with sudden failure states that need to be handled not to compromise the service itself or its security.

The **open interfaces** principle attempts to spark competition and drive innovation by recommending a public specification of the interfaces, which should be open to community definition (avoiding vendor lock-in). The recommendation favors doing standard things in standard ways while not voiding proprietary extensions. For this proposal, the openness of the interfaces allows for more controls over the existing parts of the system, with openly defined expected outcomes that may be verifiable by our solution.

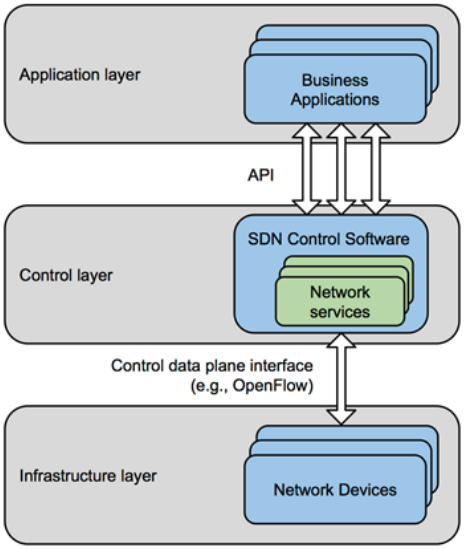
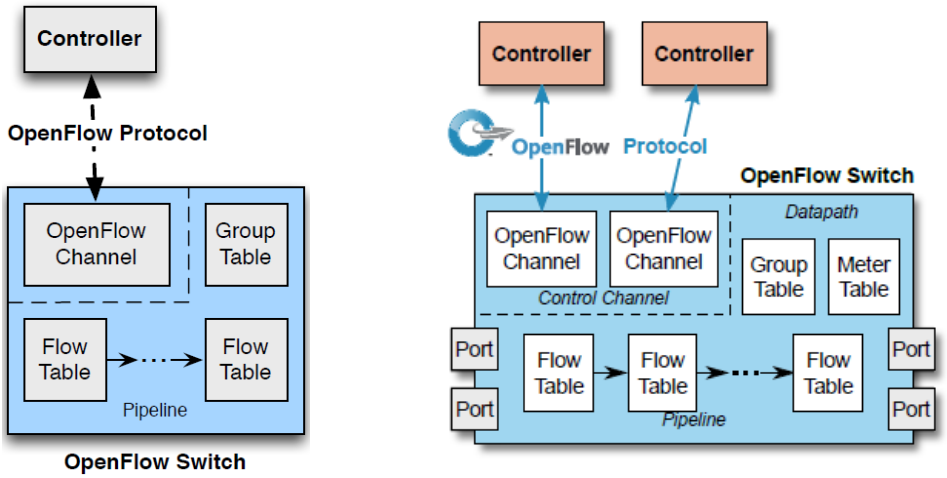


Figure 3.1: SDN overview (taken from ONF)

Figure 3.1 shows a layered overview in which these four principles are put to practice. The separation between the infrastructure and control layers follows the principle of decoupling data from the control. Unlike the multiple interfaces from the application layer to the control layer, the connection of the control layer to the infrastructure layer is made with a single arrow, highlighting the principle of logically centralized control. There are multiple network services inside the SDN controller, illustrating where the programmable network services are deployed and their scope of action. Lastly, open interfaces power the business applications in the application layer, as the interconnection between the control and the infrastructure.

The three layers are also separate domains with different mechanisms and capabilities to consider when designing a dynamic network security solution. The infrastructure layer is where policy enforcement happens. In turn, the business application layer is where policies and requirements are defined. Then, the control layer performs the decisions to enforce policies and requirements in the infrastructure



(a) version 1.0 to v1.4 [42]

(b) version 1.5 [43]

Figure 3.2: OpenFlow switch

layer.

3.1.1.1 First Generation (OpenFlow)

The communication between the control and infrastructure layers is made through well-defined protocols, following the open interfaces principle. Most notably, the OpenFlow protocol [41] is ONF's proposal to fulfill the duties of this interface and the *de facto* SDN standard. The OpenFlow protocol defines the messages, structures and capabilities that the infrastructure must be support to be compliant. There are multiple versions of the protocol, with major revisions ranging from v1.0 to v1.5. The revisions introduce new messages, which add new capabilities such as new match fields or actions.

Figure 3.2 shows the basic architecture of an OpenFlow switch. The OpenFlow channel is where control actions are done over the forwarding plane. The flow table is where such control actions are stored in the switch. The group table allows to specify control actions that multiple flow rules will match, the pipeline (Figure 3.3) is where the enforcement happens, and the ports where flows enter/exit the switch. The meter table was introduced later to implement simple Quality of Service (QoS) functions. In OpenFlow v1.5, a mechanism for controller fallback was introduced (Figure 3.2b), which allows controller redundancy to be implemented at the application level (instead of a network level, as seen in Figure 3.2a). Nevertheless, the controller is still a single element with logical central control, meaning that the fallback controller of OpenFlow v1.5 needs to be in perfect synchronization with the primary controller.

The OpenFlow protocol is very relevant to this proposal because it may determine our dynamic security system's most basic flow selection capabilities, along with the actions immediately available at the infrastructure layer. The actions have a similar scope to the available matches, including switch ports, all header data of OSI Layers 2 & 3, plus the layer four of TCP and UDP, and VLAN.

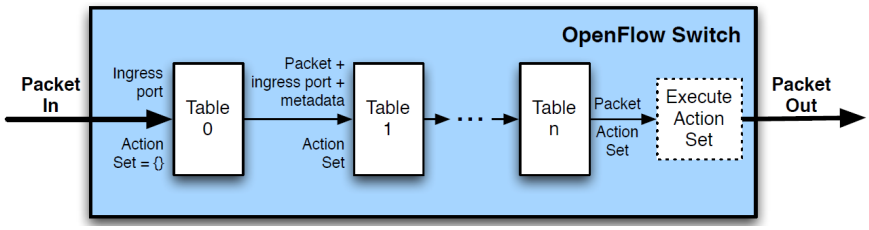


Figure 3.3: OpenFlow pipeline [42]

3.1.1.2 Next Generation (P4)

Programming Protocol-independent Packet Processors (P4) is a next generation protocol to approach SDN, geared towards lower-level programming of the integrated circuits that make up the forwarding plane. This programming allows for baking functions directly into the forwarding plane so that the control is not limited to a predefined set of actions. Because of this, P4 is more flexible than previous protocols, such as OpenFlow. Compared to programming other low-level packet processing systems, the P4 language can express more complicated algorithms in a hardware-independent way. P4 also abstracts the mapping and management of resources within the switch. Software engineering is made easier with type-checking, code reuse, and information hiding. Component libraries can feature hardware-specific features, accessible through portable high-level P4 constructs. Therefore, hardware evolution is decoupled from specific software, and debugging can be made more accessible through software models of the hardware architecture.

The described benefits already incorporate the latest features of the $P4_{16}$ language version. $P4_{16}$ breaks away from $P4_{14}$ significantly, as shown in Figure 3.4. Most notably, $P4_{16}$ splits most of the feature-set of $P4_{14}$ into different libraries, allowing for hardware-specific optimizations and further evolution of the language without breaking basic compatibility with the core library. Unfortunately, some of the functionality in $P4_{14}$ is no longer available in the core libraries of $P4_{16}$, which sets the precedent of breaking backward compatibility.

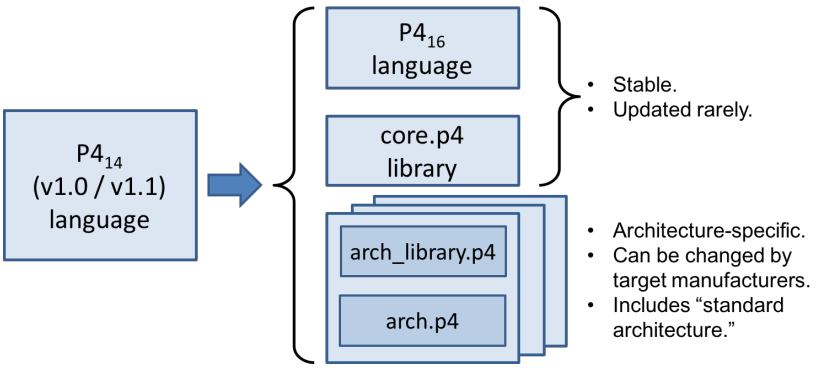


Figure 3.4: P4 language evolution (*p4.org*)

The Very Simple Switch reference architecture (as shown in Figure 3.5) allows us to see the pipeline similarities between OpenFlow and P4 programmable switches. P4 could be used to create a fully compliant OpenFlow switch, potentially getting the best of both worlds: a fully programmable forwarding plane that can extend fully supported *de facto* standard messages and features. Barefoot Networks has already released such a switch (P4 program) under the Apache 2.0 license.

However, the P4 language is driven chiefly to solve hardware development and lower-level forwarding plane development problems. Therefore, the use of P4 assumes permission to program physical devices

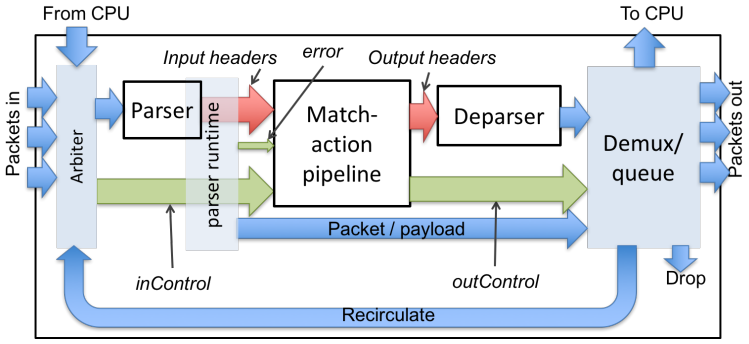


Figure 3.5: Very Simple Switch in P4 (*p4.org*)

at low level, which may be a strong assumption for tenant networks in shared infrastructure. While this proposal could benefit from the extended features of P4, there is also limited adoption of the language, with only a selection of hardware manufacturers selling supported hardware. Furthermore, for this thesis, the potential gains in flexibility and performance may not overcome the increased hardware cost and development effort compared to incorporating a tailored VNF.

3.1.1.3 SDN Security

The ONF set a baseline of security principles and practices for SDN networks [44] which explored the security model of ONF's OpenFlow protocol and the general security issues of an SDN architecture. In this document, ONF identified four significant types of SDN-specific challenges. The first is the central control, which is a single high-valued target to an attacker. Compromising the controller leads to the complete subversion of the whole SDN data plane. The second challenge is programmability. For instance, programmable interfaces exposed to other entities may be used to subvert traffic and resource isolation. Furthermore, there needs to be trust between the third-party apps and the controller. Lastly, the logically centralized controller may actually have a multi-layered implementation or other controllers in tandem (*e.g.,* redundancy), which need to communicate sensitive information about the network through their interfaces. Therefore, a compromise of one of those interfaces may lead to extensive damage in the data plane. The third challenge is integrating with legacy protocols (for compatibility reasons). It may not be easy to retrofit the desired security capabilities, so their fragilities are not propagated to the SDN control. The fourth and last challenge is cross-domain connections, which require different interconnecting controllers of different infrastructures while still enforcing a consistent policy.

Scott-Hayward et al. [45] surveyed the state-of-the-art of SDN security. Besides the challenges already identified by ONF, it was also made clear the need for detecting network errors (*i.e.,* when the switches are not enforcing the expected controller rules, or the controller app goes into an inconsistent state). *Canini et al.* [46] proposed NICE, a symbolic execution approach to test OpenFlow applications' correctness. *Son et al.* [47] introduce FLOVER, which uses modulo theory and assertion sets to verify flow policies. A common trait of the surveyed works was the lack of "real-time" processing to avoid damages to the network. We expect to address this limitation in our proposal.

3.1.2 Network Functions Virtualization (NFV)

Network Functions Virtualization introduced cloud computing capabilities to the network, virtualizing network functions and networks alike, allowing for greater flexibility, lower time-to-market, and ways to reduce the CAPEX and OPEX of operators. The ETSI specifies an NFV framework [7] which

normalizes the requirements, interfaces, and description of virtualized NFs. Figure 3.6 shows the normalized ETSI NFV architecture, which is made of four big components. The OSS/BSS is the existing operator support system, which interconnects with the new three components. The NFV MANO introduces the elements analogous to the existing operator support systems (but for NFV), allowing for complex automatic, self-provisioned, and on-demand NFV-NSs (as normalized further in ETSI NFV-MANO [48]). Then, we have the NFV Infrastructure (NFVI), which represents all the physical and virtualized hardware which the VIM will pool. Lastly, we have the VNFs themselves, with an Element Manager (EM), which allows the VNF Manager (VNFM) to perform actions within the function.

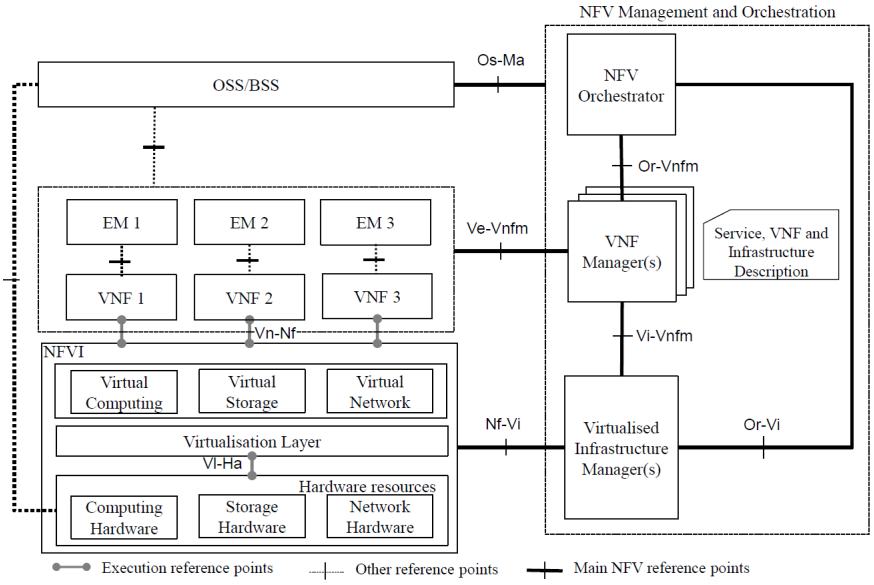


Figure 3.6: ETSI NFV architecture [7]

VNFs can be of any kind or service, from end-user applications to middleboxes. When used together with SDN, VNFs can extend the forwarding plane functionality beyond the standardized actions available in the messaging protocol. Middleboxes [49] are NFs that process flows, in ways that exceed just packet forwarding, despite not being one of the end-points of the flow. Typical types of middleboxes include firewalls, IDS/IPS, NAT, connection optimizers, and load balancers. Because middleboxes violate the end-to-end principle of network design, they can also become a security concern. Business data, privacy-sensitive information, and other classified information may be abused or leaked by a middlebox.

Furthermore, as some middleboxes act as MITM, end-point trust management and validation of the trust chain may be overridden by the middlebox, which constitutes a severe point of failure. *Naylor et al.* proposed a family of multi-key TLS protocols specifically for middleboxes to address this, first the mcTLS [50], and then the mbTLS [51]. The different keys could then be used to control the permissions over the flow inside the middlebox.

Service Function Chaining (SFC) is a network feature that makes the use of middleboxes easier through a controlled environment in which the classification, traffic steering, concatenation of the NFs, along with the respective flow context, may be passed along each function of the chain.

ETSI surveyed the potential areas of security concern across the VNF life-cycle [52], identifying ten potential concerns. The first is shared with SDN, which is the need to validate topology and the proper enforcement of control actions (an objective of this proposal). The second concerns about

the availability of management interfaces, particularly assuring that when everything fails (such as a compute node crashes), there is still a way to remotely recover it (such as a segregated network with access to the server board's remote management). The third is a secure boot, attesting to the integrity of the boot process and the validity of the boot image. The fourth is a secure crash so that if a VNF crashes, then all its open handles and transient information cannot be reused. The fifth is performance isolation so that proper QoS can be maintained, side-channels are mitigated, and availability is not compromised. The sixth is tenant authentication, authorization, and accounting, which is critical for identifying the users who can perform a given action within a given limit. The seventh is an authenticated time service, which thwarts time-skew attack techniques, and greatly benefits event log correlation. The eighth identifies the risk that some images may have hardcoded private keys (instead of proper key distribution). Thus, the functions become exposed to attacks once the key is compromised (making it harder to recover smoothly, given the lack of proper key distribution infrastructure). The ninth is the concern about backdoors caused by the hypervisor's test and monitoring capabilities and similar features available in the running hardware. Finally, the tenth concern is about multi-administrator isolation, mainly due to the Lawful Interception (LI) requirements that telecoms are usually bound to comply with.

3.1.3 Network Slicing

Network slicing is an emerging trend that builds upon SDN and NFV to split a physical networking infrastructure into many virtual networks (slices) suited to the needs of their tenants. Each slice has unique isolation and assurances according to the set of business KPIs, shown in Figure 3.7.

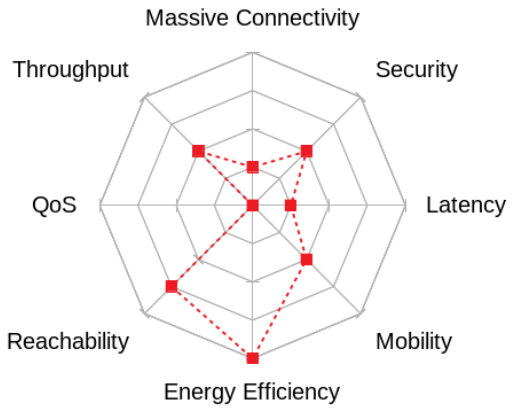


Figure 3.7: Slice KPIs (GSM Alliance [53])

Isolation is a fundamental feature of Network Slicing. The better the isolation, the more reliable (*i.e.,* predicatable) the slicing solution is. A "slicing system" that can only ever have a single slice is a regular non-sliced network, *i.e.,* an already well-researched topic. Therefore, by definition, multiple slices will (at some point) have to coexist by sharing the same infrastructure. This coexistence is determined by the minimum requirements set for each slice. As long as these are delivered, interference does not occur, therefore preserving isolation. Defining what constitutes interference for that slice, setting the minimum requirements, and enforcing them is the critical part of security in network slicing.

The Next Generation Mobile Networks Alliance (NGMN) 5G security recommendations for Network Slicing [54] identified ten key issues. Controlling inter-Network Slices communications is a fundamental to ensure isolation. Instantiation time impersonation attacks against Network Slice Manager or host (physical) platforms within an operator network, impersonation attacks against a Network Slice instance

within an operator network, and impersonation attacks against different Network Slice Managers within an operator network would undermine the system's integrity. Different security protocols or policies in different slices may result in hard-to-solve arbitration issues. Denial of service to other slices or exhaustion of security resources in other slices are risks that must be mitigated/solved. Side-channel attacks across slices may break isolation. Hybrid deployment models raise challenges in maintaining same service levels and isolation. Lastly, the sealing between slices when the UE is attached to several slices must also be considered.

Similarly, 3GPP's *TR 33.811* [55] study of network slicing security for 5G identifies four issues. Unauthorized access to Management Exposure Interface yields great power over the system. Protecting the results of NSI supervision/reporting is crucial for preserving confidentiality when handling classified data. Protecting Network Slice Subnet Template is fundamental for system integrity. Lastly, an insecure procedure for capability negotiation could undermine the whole system.

Further recommendations in underlying technologies were also considered, such as ETSI's for NFV [52], which surveys the potential areas of security concern across the VNF life-cycle. Similarly, ETSI NFV-SEC also covers essential aspects of the ONF SDN security whitepaper [44].

Kotulski *et al.* [56] state that it is necessary to identify isolation attributes, create a kind of abstraction layer to assure end-to-end isolation on a particular strength level, and introduce adequate security policies. The authors conclude in their survey that no common description of isolation capabilities could be used for automatic deployment. Therefore, it is essential to define the expected initial isolation level (*e.g.,* security) and design dynamic isolation mechanisms that can enforce the isolation level of any given service.

Each slice has its isolation constraints, determined by its KPIs. Therefore, interference can be broadly defined as anything that breaks the ability to deliver the KPIs of a given slice. Such interference can be attributed to different origins. For instance, the KPIs may have been poorly selected for the slice application, the service provisioning may be insufficient to deliver those security-related KPIs, or there may be adversaries disrupting services. Consequently, an effective network slicing solution needs to address management, performance, and security as a whole.

The Network Slice Manager (NSM) must be able to track flows and function interactions across slices within their administrative domains. The NSM is responsible for the abstracted virtual network and interacting functions within the slice. However, the same NSM is not responsible for the orchestration or correctness of services provided by the slices. Secure network slicing solutions must ensure the main security principles, traditionally categorized into the Confidentiality, Integrity, and Availability (CIA) triad. In such a context, when associated with network slicing, these principles translate to:

1. **Confidentiality**: must ensure that packets are not made available outside of the slice that generated them or the slices allowed to interconnect. Additionally, the data carried in those packets or held within the NFs that process them must not be available to anyone other than the authorized elements or end-users.
2. **Integrity**: the system must not be subverted, either by tampering with data or by replacing its functionality. In other words, only Slice-Owners may change the application part of their NFs, define the flow processing within the slices, or change the inter-slice configurations. Cross-talk between slices cannot be allowed. Inter-slice communications must only happen through their respective interfaces.
3. **Availability**: the system must be reachable and working as expected when it is required. This reachability means that the NSM and NFs must remain accessible at all times. At the same time, the slices and application part of the NFs must be accessible as long as the contracted

infrastructure resources are not exceeded. Another important aspect is the processing and response times, which must remain under the threshold as specified in the Service Level Agreement.

3.2 Moving Target Defense (MTD)

MTD is a well-researched field that spans over five different areas: Dynamic Networks, Dynamic Platforms, Dynamic Runtime Environments, Dynamic Software, and Dynamic Data. In the next sections, we will present the works most relevant to this proposal in each area.

3.2.1 Mechanisms for Dynamic Networks

The mechanisms for dynamic networks can be grouped under four categories: IP address and port mutation, resolved path mutation, fingerprint mutation, and multiple mutations. Later works feature multiple mutations, while earlier works researched mostly single mutations. Given the importance that networking has for this proposal, we will start by introducing each mutation category to provide a foundational background about the expected capabilities of the individual mechanisms. Then, we will present the works that perform multiple mutations, which better reflect the state of the art of systems closer to this proposal. Finally, we will approach the works which address the MTD system optimization, evaluation, and strategy selection challenges.

3.2.1.1 IP Address and Port Mutation

The IP address and port mutation rely on shuffling end-point information to increase the effort required to perform a network service attack. This mutation is effective in threat models where the attacker is not a possible legitimate client (*e.g.,* attempting to act in another capacity), therefore lacking an easy means to discover how to reach the service is a notable security improvement. The first research predates the creation of MTD, having *Kewley et al.* [57] proposed Dynamic Network Address Translation (DYNAT), a transformation function of the destination IP address and port that used an encryption algorithm (RC5 [58] due to matching the ciphertext output size), along with a secret seed and a time-based secret key. This transformation meant that only those who held the shared secrets could reliably contact the services behind DYNAT. Due to the properties of RC5 [58], those who did not would be misdirected to a random IP address and port pair (that directly resulted from a transformation with the wrong secrets). This misdirection would make network scans take longer while still being harder to return actionable information, aid the detection of attacks such as DoS (higher traffic dispersion than the limited valid destinations), and put a time limit on a session hijack. A more modern approach than DYNAT was presented by *Jafarian et al.* [59], introducing the OpenFlow Random Host Mutation (OF-RHM) technique. Unlike DYNAT, there is no need to pre-share secrets to find the end-point, as new DNS responses are created according to the current mutation. A random virtual IP address will be assigned from a pool of available addresses, being this random function either uniform or weighed (according to the chosen mutation type). The service end-point will have an actual IP address that does not move, being the random virtual IP address a moving translation to the matching real IP of the service. Previous proposals, such as Network Address Space Randomization (NASR) by *Antonatos et al.* [60], allow mutating the IP address of the service by changing the responses in the DHCP protocol. *Luo et al.* [61][62][21] proposed, and incrementally improved, a mechanism that also allowed for port-hopping (on top of IP mutation). The most significant difference between the first two proposals was in the technical approach. TAP-based Port and Address Hopping (TPAH) [61] relied on TAP virtual devices to perform the hopping, while Random Port and

Address Hopping (RPAH) [62] adopted a higher level Hopping Gateway approach. Keyed-Hashing based Self-Synchronization (KHSS) [21] was proposed to address the synchronization issues between the end-points when network events such as packet re-ordering, packet loss, retransmission, and others happened. The KHSS proposal did not scale well, demanding too much overhead at higher network throughputs. This issue prompted *Liu et al.* [63] to devise an address hopping method that did not require a new synchronization protocol but instead extended the DHCP protocol for that purpose. Unlike the previous methods, which are direct translations of serviceable end-points, *Sun et al.* [64] proposed Decoy-Enhanced Seamless Network IP Randomization (DESIR), which on top of the previously discussed mutations, introduced deception as a means to frustrate attackers.

3.2.1.2 Resolved Path Mutation

The resolved path mutation relies on changing the network paths taken by flows to prevent eavesdropping, other reconnaissance techniques, and DoS attacks. There are two fundamentally different approaches to resolving path mutation. The first takes the network topology graph and focuses on the unique connections (or routes) available in-between nodes. In contrast, the second favors the unique forwarding nodes over the connecting latices. Most of the current works fall within the first approach, particularly a Dynamic MultiPath Routing[1] technique, which is adapted to fit the MTD objectives. *Duan et al.* [65] proposed the Random Route Mutation (RRM) mechanism modeled using Satisfiability Modulo Theory (SMT) to identify the available route mutations. Then, the same team (but with a different first author) showed the suitability of this approach using game theory against persistent attackers [66]. *Rauf et al.* [67] argue that, while previous works are suitable for Local Area Networks (LANs), they are inadequate to defend Wide Area Networks (WANs) such as the Internet infrastructure. It is unfeasible that a single entity would have sufficient control over Internet routes to mutate them between the two end-points. They proposed the End Point Route Mutation (EPRM) method, which considers the end-points as if they were routing nodes. Unfortunately, the author evaluated the time complexity of route selection using the EPRM method as being close to a third-order polynomial function. *Dolev et al.* [68] used an *n-k secret sharing scheme* and Max Flow Theory to create SDN-based private interconnections. This scheme allows bounding the time complexity by a second-order polynomial, an improvement over EPRM, having the number of nodes as a quadratic variable.

3.2.1.3 Fingerprint Mutation

Fingerprint mutation changes some flow patterns so that extracting the right features to identify vulnerable services correctly becomes much more challenging. There are two fundamentally different approaches to fingerprint mutation. The first is called "direct mutation," which replaces the identifying features by random data, making the attacker's feature extraction process run continuously without ever getting a verifiable match. *Kampanakis et al.* [69] proposed an SDN-based MTD technique, which drops requests or sends random payloads to end-points deemed illegitimate. *Zhao et al.* [70] devised an SDN-based Fingerprint Hopping (FPH) method modeled after a signal game. Therefore an appropriate hopping strategy is achievable by reaching the equilibrium(s) of the game. The second approach is deception, which instead misleads the attacker into identifying a false fingerprint, making it waste time and resources pursuing either an inexistent target or a decoy system. *Al-Shaer et al.* [71] proposed Mutable Networks (MUTE), an architecture that employs two anti-fingerprinting techniques, session-control messages interception and modification, or firewall-level false positives generation. *Albanese et*

[1] `https://tools.ietf.org/id/draft-kapoor-rtgwg-dynamic-multipath-routing-01.html`

al. [72][73] incrementally devised deception mechanisms that returned to the attacker believable (but wrong) fingerprints. Their latest work focused on reducing the distance between the actual view and the misleading view, changing the critical information necessary to successfully conduct the attack while simultaneously incrementing the likelihood of the attacker believing the forged information. *Achleitner et al.* [74] presented Reconnaissance Deception System (RDS), which uses SDN and places known vulnerable hosts in the network(s), thus defending against insider threats of an organization. *Shi et al.* [75] proposed CHAOS, an SDN MTD system that employed rapid obfuscation of real servers and deployed clusters of decoys to which suspicious traffic would be redirected. In order to do so, all traffic would first go through an IDS. A common requirement to most fingerprint mutation solutions is that a proper illegitimate usage detection function exists in order to prevent disrupting legitimate users.

3.2.1.4 Multiple Mutations

Later works coordinate movement using more than just one type of mutation, creating more uncertainty for the attacker. On top of fingerprint mutation, the MUTE [71] architecture also allows mutating the IP addresses. MUTE's early work in network reconfiguration for MTD would become a reference for the later works which leveraged emerging network configuration technologies. Proposals such as SDN-MTD [69] and CHAOS [75] used SDN's abilities to mutate IP addresses, ports, and fingerprints. The SDN-based Double Hopping Communication (DHC) [76] can mutate the resolved path, as well as IP addresses and ports.

Aydeger et al. [20] proposed an MTD framework for ISP networks using SDN and NFV, which has Virtual Collection Points (VCPs) to store interesting traffic so that forensic analysis can be performed later. The framework uses Virtual Shadow Networks (VSNs) with Virtual Shadow Hosts (VSHs) for honeynet-like behavior and Virtual Shadow Nodes to perform route mutations. The authors evaluated their framework using a Mininet Proof of Concept (PoC), in which they determined their solution to be suitable to scale to an ISP size (algorithm complexity of $O(n \times log(n))$).

3.2.1.5 Evaluation, Strategy, and Optimization

An expected significant contribution of this proposal is the ability to dynamically adjust the security level of any given network service. To do so, we must first evaluate what constitutes a given level of security, define the strategies and understand their outcomes security-wise, and optimize the solution having specific parameters as constraints.

In order to evaluate security, *Connell et al.* [77] created a framework that attempted to quantify the utility of a movement in any MTD system in a probabilistic manner. The framework had a few limitations, including crude probabilistic estimations, simple cost modeling, and a utility function that did not accept zero-rating weaknesses. Some limitations were addressed in a later work [78], in which the cost and performance modeling of MTDs were improved.

Liang et al. [79] surveyed how game theory can be applied to network security, in particular, to model attack-defense games into different types of networks. *Liang et al.* classify the attack-defense games using three parameters: the number of stages, the availability of information about previous states, and the knowledge of how both the attacker and defender deliberate their next move. Using this classification, the authors referenced the existing works better suited to that combination of parameters (*i.e.,* yield the best strategy). The significant limitations were that most strategies did not contemplate more than two players (a single attacker against one defender). Furthermore, the criteria to set the

utility (payoff) functions lacked further research. Additionally, the models that relied on a transition probability (stochastic games) were made under the assumption of a finite number of states.

Connell et al. [23] researched maximizing the utility function to optimize their MTD solution considering the limits in the reconfiguration capabilities of the infrastructure. The utility function factors both response time and the probability of a successful attack. The limitation of this work is the assumption that all resources are "homogeneous," that is, they have the same reconfiguration capacity as well as processing power. *Van Leeuwen et al.* [80] had already researched the operational costs of deploying an MTD solution over different types of NFs, ultimately proposing a defensive work factor approach to quantify the impacts over implementation, performance, stability, and effectiveness.

Connell et al. [81] argued that no single MTD solution provided defense against all types of attacks, therefore successfully performed an analysis of how concurrent MTD solutions could improve the defensive effectiveness. The analysis also proposes ways to predict the cumulative effects of different kinds of MTD, especially regarding effectiveness and performance impact. Similarly, *Zhao et al.* [82] proposed another model to analyze the effectiveness of MTD.

Lakshminarayana et al. [83] conducted a CBA of MTD applied to power grids, particularly for false data injection attacks against state estimation of power lines, under the assumption that a bad data detector is already deployed in the state estimation process. The research focuses on the properties of power systems, namely, the MTD mutation is done over the reactance of the power line. Such framing entails a loss of generality for common computer science problems due to the constrained optimization problem within the power grid model and an optimal power flow problem. Nevertheless, the work points out the need to perform a CBA to assess MTD as a viable solution, highlighting how previous works did not address this issue but assumed a low enough operational cost would be achievable. *Villarreal-Vasquez et al.* [84] demonstrated an MTD-based approach to increase the resilience of cloud systems, mitigating attacks and failures of critical functions. These may be of particular relevance in the context of 5G, a softwarized and virtualized network.

3.2.2 Mechanisms for Dynamic Platform

Dynamic platform changes the properties of computational resources, according to the service, the attack domains, and attack mechanisms against which must be defended. The properties changed can be the CPU vendor/micro-architecture, OS version, platform data format, and more. For instance, if we were defending against speculative executions side-channel like attacks, the platform is the hardware in which our service is being run. *Moon et al.* [85] proposed *Nomad*, a side-channel mitigation technique that leverages provider-assisted Virtual Machine (VM) migration to minimize co-residency time, preventing information leakage even against unknown side-channel attacks. The movement decisions are made using an Information Leakage model (InfoLeak), determining how the placement algorithm will reshuffle the VMs across the provider.

However, another definition of the platform may be the Operating System (OS), which runs our service. Therefore, *Thompson et al.* [86] proposed the MORE approach, which rotates the OS. MORE protects against the exploitation of vulnerabilities in the OS, not the ones in the service itself. The core mechanism is achieved with a load-balancer as front-end and a pool of VMs with the same service (installed in different OSs) in the back-end. Periodically, the active VMs are replaced by inactive ones to be inspected for intrusion attempts, and their state can be rolled back to the last known good configuration.

Furthermore, with the rise of web applications and web-based information systems, the web server application may also be seen as the platform that enables those systems. Therefore, *Thompson et*

al. [87] later proposed DARE, which employed a strategy similar to the MORE approach, this time with the web server application being rotated (instead of the OS).

3.2.3 Mechanisms for Dynamic Runtime Environment

The mechanisms for dynamic runtime environment create, inside the NF, a different execution environment each time the program is started. These mechanisms can be separated into two major categories, the first performs address space randomization, while the second does Instruction Set Randomization (ISR).

The first technique is already widely used. Address Space Layout Randomization (ASLR) consists of adding a random offset to the base pointers of each section of the memory layout to difficult the control-flow hijack through a buffer-overflow attack. Each modern OS implements ASLR, such as Linux [88] or Microsoft Windows (after Vista). ASLR has some limitations [89], such as being dependent on the level of entropy available in the system. *Bittau et al.* [90] demonstrated that the mechanism may still be bypassed even with high entropy.

The second technique, ISR, is less commonly used but effective to thwart binary code injection to hijack the control flow. The main idea behind ISR is that the rogue instructions an attacker injects into the running program will not result in the predicted outcome but rather an unactionable result that depends on the current meaning of that code [91]. In practice, ISR introduces higher overheads than just ASLR [92] due to the code translation needs (which is often done through emulation). *Jiang et al.* [93] proposed RandSys, which randomized the system call instructions in conjunction with an ASLR technique, combining the defensive benefits of having both the used addresses and system call instructions uncertain to the attacker. Later works, such as *Kim et al.* [94], proposed hardware-accelerated instruction-set transliteration.

3.2.4 Mechanisms for Dynamic Software

Dynamic Software changes the running code on the fly, modifying the instructions, order, grouping, or format. These modifications are made at the program level, independently of the techniques employed by the runtime environment. For our purposes, the techniques may be helpful in containerized environments, where the user has limited administrative power over an essential part of the runtime environment. *Chao Zhang et al.* [95] proposed Compact Control Flow Integrity and Randomization (Compact Control Flow Integrity and Randomization), which successfully address the control-flow hijacking attacks from within the application. *Koning et al.* [96] introduced MvArmor, an anti-hijacking approach that uses hardware-assisted virtualization (Intel VT-x) to protect the program's address space and subsequent control-flow hijack.

3.2.5 Mechanisms for Dynamic Data

The foundational work of *Ammann et al.* [97] introduced a data diversity technique that increased the fault tolerance of an application by transcribing the input to a functional equivalent which should produce the same outcome on success. Failures are highlighted by their unexpected outcomes, which incidentally also makes the abuse by an attacker harder. *Christodorescu et al.* [98] later applied similar ideas to protect Internet services, which directly relates to the subjects of our proposal.

3.2.6 Cyber Mimic Defense (CMD)

The CMD concept was first introduced by *W. Jiangxing* [99]. However, as his publication was written mainly in Mandarin, the CMD framework presented by *Hu et al.* [100] and the Mimic Network Operating System (MNOS) for SDN (also by *Hu et al.* [101]) were used as the reference for this subsection. It should be noted that *W. Jiangxing* is a co-author of both publications.

CMD is founded under two strong assumptions (taken as axioms):

A1 The same goal can be reached by multiple algorithms, perhaps with different performance, but functionally equivalent.

A2 Different implementations will have different mistakes that will not generate the same faulty outcome at the same place.

The core architecture of CMD is the Dynamic Heterogeneous Redundancy (DHR) model, as depicted in Figure 3.8. From the get-go, a limitation of this model is that CMD can protect only the functions that obey the input-output model. That is, having the same input parameters, the function must consistently return the same output. The Input Agent will receive the function's input and send it to the current set of executive entities. Each of the current executive entities will generate its output and send it to the Arbitrator. Then, using similar consensus functions as in distributed systems theory, the Arbitrator will determine the correct output and send it to the original requester. It must be noted that the set of executive entities is dynamic and may change over time. The set is initially populated at random by a dynamic selection algorithm, which will take a given number of components from the pool of heterogeneous components. Then, as the Arbitrator determines that a component has yielded a faulty output, that component should be replaced by another from the pool of heterogeneous components.

Therefore, as long as (1) the pool of heterogeneous components have sufficient implementation diversity, (2) all components in the pool are functionally correct, (3) the set of executive entities is selected unpredictably, and (4) a sufficient number of entities is running for the Arbitrator to deliberate, then the DHR model should assure correctness and defense against uncertain threats.

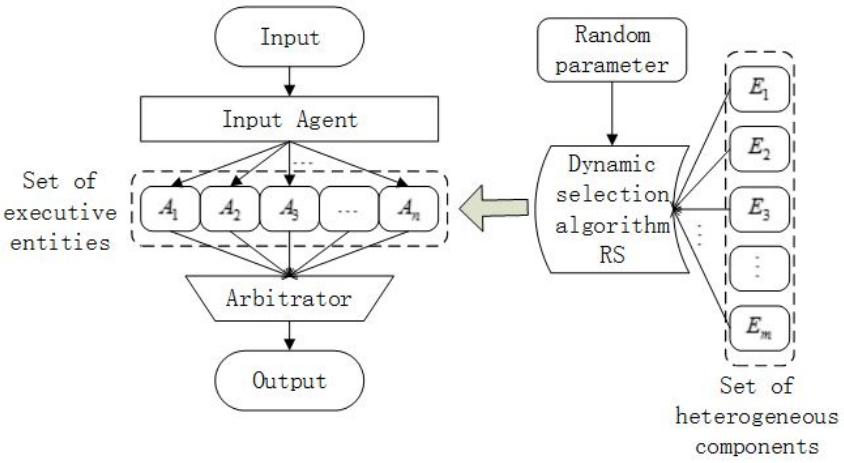


Figure 3.8: The DHR model *(Wikimedia version, translated from the original [99])*

For our purposes, this means that we may be able to address the correct control enforcement in the forwarding path using the DHR model, with necessary modification to make the inherited race conditions of the output comply with the input-output model. Additionally, we may assure the

correctness of given NFs assuming a sufficient pool of diversity exists and the NF complies with the input-output model.

CMD is still in its early years of research; therefore, only a few works have been published so far. *Hu et al.* [100] proposed a framework for CMD in which the DHR model was presented and evaluated using a combination of analytical, simulation, and PoC methods. The chosen PoC was with non-virtualized Consumer Off The Shelf (COTS) routers, and it did not thoroughly verify the DHR's heterogeneity requirements (as pointed by the authors). Another work by *Hu et al.* [101] showed CMD and the DHR model applied to protect the SDN controller in a proposal named MNOS. Once again, the authors conducted a combination of analytical, simulation, and PoC methods. The PoC results showed the ability to prevent the insertion of an abnormal flow, while the analytical and simulation evaluation justified the soundness of the approach.

Ma et al. [102] researched how redundancy affects CMD systems, particularly when subjected to DoS attacks. This choice is significant because the original authors of the DHR model had set the DoS type of attacks as a delimitation of the model. Therefore, the research on how to thwart this kind of attack effectively is still lacking. *Ma et al.* work showed that increasing redundancy is an effective strategy against DoS attacks targeting CMD systems.

Liu et al. [103] proposed a framework to detect and respond to zero-day attacks using a CMD architecture. The framework detects attacks using the abnormal outputs within the traditional DHR model, then responds to the attack with deception mechanisms (honeynets), more traditional responses such as IP blocking, or ultimately patching the newly found vulnerability. The authors conclude that the framework is a complete defense method, which does not depend on external rules or filters to be effective. However, the authors also note that the framework was in its early days, pointing out the need for further research.

3.3 Summary

Telecom operation requires remote management actions across different locations and parties. Softwarized and virtualized networks in the Telecom world are no exception. Therefore, network interfaces must expose methods to control the network and other systems through an underlying communication medium. These interfaces may be exposed within the same administrative domain (i.e., single-party across different locations) or may cross administrative domains (i.e., multiple parties working in an inter-domain deployment). Integrity and availability are critical for proper operation within the CIA triad, while confidentiality is desirable. Further considerations about risk and exposure will be in the threat model. Achieving a working network slicing solution requires defining the regular operation of the underlying network virtualization and softwarization technologies. Those actions fulfill the requirements for the virtualized entities that, together with a programmable network forwarding plane, deliver the logical grouping and assurances that fulfill the requested Network Slice (NS).

While addressing the threats identified in the related works (particularly the Network Slicing Subsection 3.1.3), it became evident that most threats require active interaction with the target systems. Solving these threats would require a deep analysis of the affected systems and implementing adequate controls. The mechanisms envisioned in the thesis motivation can provide an additional layer of protection and mitigate the failure of one of these controls. That includes protecting against zero-day vulnerabilities found in any of the protected interfaces or even defending against some insider threats.

Actively probing the system should become much more demanding. The existing related works

already provide some approaches that could be explored to defend our network slicing system. However, there are fundamental limitations that set the ground for enhancements that are required to have a practical network slicing system. In particular, the security enforcement mechanisms must not disrupt the NS core KPIs or introduce significant management challenges (such as an additional synchronization protocol or ephemeral keys that may not be recoverable for a future system audit/*post-mortem*).

The security mechanisms should be integrated concisely to provide additional response capabilities once the malicious activity is detected. The mutation cost (*i.e.,* computational cost) must not negate the system's utility. Therefore, the contributions must be packaged for orchestration within the ETSI-NFV framework mindset, enabling their use in the network slicing superset.

CHAPTER 4

MTD Approach

This chapter introduces an effective MTD approach that fulfills the needs identified in the related works chapter (Chapter 3). The chapter is structured as follows: Section 4.1 performs a brief introduction. In Section 4.2, we introduce the relevant background and related works. Then, in Section 4.3, we detail our proposed MTD mechanism. In Section 4.4, we critically evaluate the results from a PoC. Finally, in Section 4.5, we present the conclusions, significant findings, and future work.

4.1 Introduction

The tighter integration of business verticals into previously core network operations, as in the 5G Vertical Slicing use-cases, raises significant security, trustworthiness, and reliability issues [104]–[106]. A single business vertical must not compromise the integrity of the platform that powers all other verticals. Similarly, the vertical applications that are entrusted to the platform should not be compromised by external parties. Cloud Computing, and by extent Edge Computing, encourage a de-facto standardization of the same protocol stack (TCP/IP) [12] across previously segregated systems. The use of standardized protocols, and the reuse of similar software solutions, substantially lowers the cost of an attack, as one vulnerability within a component of this stack now covers a broader range of potential targets.

To achieve a successful attack and exploit a target, the attacker must first do a reconnaissance of the virtual surroundings, identifying the objectives, along with the vulnerabilities of each service that may lead to its goal [24]. Security vulnerabilities are mostly a particular software bug [107], and bugs tend to be revealed over time. Therefore, time is on the attackers' side to find and exploit a vulnerability, while network function maintainers have time playing against their security designs; this effect of time is known in the security context as the asymmetric relationship between attackers and defenders. The core principle of MTD is to identify the available network configuration space and dynamically explore it (*i.e.,* create movement) to frustrate the attackers' endeavors [18], [38]. Provided the attacker does not know the secret that creates the movement, the MTD mechanism may break the attacker's connectivity to the service or forge data (or interactions) so that the attacker acts upon faulty intelligence.

The practical issue with MTD is how to generate movement across peers without disrupting legitimate usage, hindering other ancillary systems (such as auditing, accounting, and billing), or being detrimental to an attack's attribution and forensics [20]. Synchronizing the movement across

peers without incurring heavy overheads (synchronization protocol), sacrificing mutation speed, or connection reliability (*i.e.,* errors due to desynchronization) is a significant challenge [21]. Furthermore, the target environment imposes the need for adding/removing peers at any time without disrupting the service or compromising the key management practices.

Our MTD mechanism is inspired by the 2FA methods already widely deployed by Internet Services [108]. On the one hand, those 2FA methods do not replace the existing account authentication mechanisms but complement them with an extra factor. The MTD approach has similar nature, adding an extra factor to the existing defense mechanisms (*e.g.,* firewalls or IDSs/IPSs). On the other hand, the 2FA methods in Internet Services already established practices that deal with peer synchronization, peer management, and key management issues [109], [110].

The most widely available type of One-Time Password (OTP) in public Internet services relies on RFC 6238 [110], also known as TOTP. Each peer calculates the temporary authentication code (OTP) using a per-peer unique pre-shared secret and the device's clock, truncating the output to a set number of digits (usually six) as shown in Figure 4.1. The peer becomes authenticated if the sent OTP matches the one calculated independently by the other peer. The TOTP method works if the secret is known between parties, the clocks are sufficiently synchronized, and the same configuration is used (*e.g.,* OTP length and expiration interval). Should one of the OTPs be disclosed, the adversary has a limited time to act upon that knowledge, as that OTP will expire.

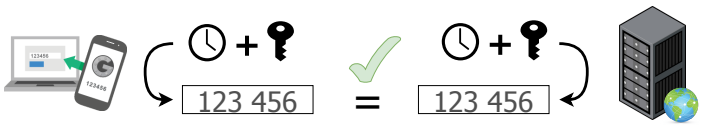


Figure 4.1: Typical use of Time-based One-Time Password (TOTP) for Two Factor Authentication (2FA)

In the following section (Section 4.2), we introduce the relevant background and related works. Then, in Section 4.3, we detail our proposed MTD mechanism. In Section 4.4, we critically evaluate the results from the PoC. Finally, in Section 4.5, we present the conclusions, significant findings, and future work.

4.2 Background and Related Work

Edge computing, 5G, and Network Slicing depend on platform management interfaces crucial to delivering their purpose: controlled on-demand Life-Cycle Management (LCM) of virtual networks and network services placed accordingly to the required network assurances (such as low-latency or high-throughput).

These platforms are usually built atop existing technologies and standards, such as the ETSI NFV [7] and SDN. The ETSI NFV architecture describes a set of MANO components and interfaces, commonly known as NFV-MANO [111]. Those MANO interfaces allow communication with internal components that wield broad power over the platform; therefore, compromising such components would have a massive impact on the platform and its services.

In Chapters 2 and 3 we have addressed the motivating framework, its concerns and techniques to secure these MANO interfaces.

We will summarize the existing MTD mechanisms, as presented in Chapter 3. The IP address and port mutation rely on shuffling endpoint information to increase the effort required to perform a network service attack. This mutation is useful to disrupt the reconnaissance and the delivery stages of the kill-chain [24]. The first research predates the discussions that lead to the creation of MTD [38].

Kewley et al. [57] proposed DYNAT, a transformation function of the destination IP address and port that used an encryption algorithm (RC5 [58] due to matching the ciphertext output size), along with a secret seed and a time-based secret key. This transformation meant that only those who held the shared secrets could reliably contact the services behind DYNAT. Those who did not, due to the properties of RC5 [58], would be misdirected to a random IP address and port pair (that directly resulted from a transformation with the wrong secrets). This misdirection would make network scans take longer while still being harder to return actionable information. It would also help detect DoS attacks (higher traffic dispersion than the limited valid destinations) and put a time limit on a session hijack. Our proposed mechanism builds upon the core concept of using time and a cryptographic replacement of existing header fields of DYNAT, using more modern and familiar methods such as TOTP instead of a direct RC5 and SDN control instead of NAT. These differences are not just a technical refresh of an older idea, but rather an essential change that copes with the new challenges of achieving a suitable mechanism within Edge computing's stringent requirements or, in particular, 5G vertical use-cases.

A more modern approach than DYNAT was presented by *Jafarian et al.* [59], introducing the OF-RHM technique. Unlike DYNAT, there is no need to pre-share secrets to find the endpoint, as new DNS responses are created according to the current mutation. A random virtual IP address will be assigned from a pool of available addresses, being this random function either uniform or weighed (according to the chosen mutation type). The service endpoint will have a real IP address that does not move, being the random virtual IP address a rolling translation to the matching real IP of the service. We acknowledge the significant contribution made by the introduction of OpenFlow in the OF-RHM technique. However, we fundamentally diverge from using the SDN controller to generate interactions (*i.e.,* DNS replies) with the authenticating participants or needing any signaling to communicate the mutations (*i.e.,* the TOTP mutations are calculated independently without needing the communication channel). NASR by *Antonatos et al.* [60] also allows mutating the service's IP address, this time using DHCP messages.

Luo et al. [61][112][21] proposed, and incrementally improved, a mechanism that also allowed for port-hopping (on top of IP mutation). The most significant difference between the first two proposals was in the technical approach, with TPAH [61] relying on TAP virtual devices to perform the hopping. In contrast, RPAH [112] adopted a higher level Hopping Gateway approach. KHSS [21] was proposed to address the synchronization issues between the endpoints due to network events such as packet reordering, packet loss, and retransmission. While *Luo et al.* KHSS proposal did not scale well, demanding too much overhead at higher network throughputs, our proposal does not require additional MTD synchronization messages in the communication channel to hinder its scalability. Furthermore, our approach aims for familiarity with already established practices (such as TOTP 2FA), leveraging the management through ETSI-NFV and NFV-MANO interfaces, and compliance with the more stringent Edge computing requirements.

The evolution of MTD mechanisms tends to introduce more functionalities rather than significant changes in the synchronization mechanism and compliance with the prior art, as is achieved with TOTP. *Sun et al.* [64] proposed DESIR, which introduced deception on top of the previously discussed mutations as a way to frustrate attackers. Later works coordinate movement using more than just one type of mutation, creating more uncertainty to the attacker. On top of fingerprint mutation, the MUTE [71] architecture also allows mutating the IP addresses. MUTE's early work in network reconfiguration for MTD would become a reference for the later works which leveraged emerging network configuration technologies. Proposals such as SDN-MTD [69] and CHAOS [75] used SDN's

abilities to mutate IP addresses, ports, and fingerprints. The SDN-based DHC [76] can mutate the resolved path, IP addresses, and ports.

In sum, our proposal addresses the most significant barriers to the adoption in Edge computing platforms and use in 5G business verticals use-cases. The design has NFV and SDN in mind, easing the deployment in such environments, allowing for the needed scalability, and taking advantage of SDN switches forwarding-plane performance. The mutation synchronization problem is mostly reduced to regular clock synchronization, a familiar problem in Edge applications.

4.3 Proposed Solution

We begin our proposal recapping its core premise. If the attacker can interact with the network socket of sensitive services, then time is on its side to find potential vulnerabilities in that service stack. For example, a remotely triggered Control-Flow hijacking vulnerability [113] could allow running arbitrary commands with elevated system permissions, thus subverting the application logic and bypassing the security controls in place. A prime example of this vulnerability class is the EternalBlue [114] exploit that subverted Microsoft Windows' SMB service. The effect of time favoring the attacker's endeavors in finding bugs/vulnerabilities that may be exploited to fulfill actions over the objectives is known in the MTD field as the asymmetric relationship between the attacker and the defender [18], [38]. To curb this disadvantage, we propose a port-mutation MTD mechanism designed to comply with the performance, management, and scalability requirements of sensitive network functions in Edge computing, 5G, and Network Slicing. The envisioned mechanism works alongside other existing controls, effectively acting as a network-bound seamless 2FA code.

We propose taking the TOTP 2FA method and using it as the movement generator function of an MTD system, with minor adjustments to mutate the sensitive NFV-NSs or private management interfaces' listening port. Much like in current Internet services, there is a pre-shared secret that authenticates a unique endpoint in the context of that service. We highlight that this approach would not add any immediate data overhead to the communications because the OTP is sent in an already existing packet header field that can be easily changed. Therefore, our MTD solution makes the NFV-NS move across the $2^{16} - 1 = 65535$ possible listening ports in a way that resembles an implicit 2FA system working seamlessly at the network level (Figure 4.2). Each peer observes a different movement accordingly to the source IP address, the pre-shared secret, and the configured mutation interval. Reaching the service's real port is a simple translation upon matching a valid OTP (*i.e.,* successful authentication).

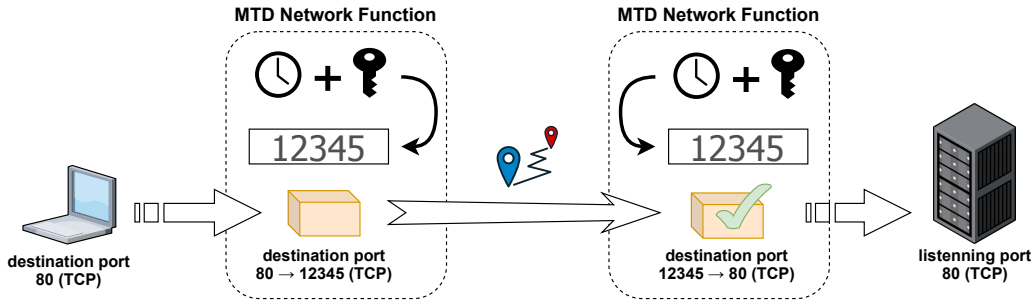


Figure 4.2: Proposed use of Time-based One-Time Password (TOTP) for the Moving Target Defense (MTD) mechanism

However, despite the straightforwardness of the idea, we must address some fundamental differences before such a mechanism could be workable. For starters, unlike the Internet Services 2FA systems,

the authentication is no longer a single discrete event at human speed but can be rather a continuously repeating action over every packet. Our MTD mechanism must cope with the volumes and velocity of automatic communications. Furthermore, the TOTP standard was designed to operate at a human timescale. The expiration intervals range from minutes to (at most) tens of seconds so that the OTP is usable in the 2FA system. Without that human element bottlenecking the system, such a prolonged steady-state in the continuous communications is detrimental and allows for exploitation (*i.e.,* movement is too slow). Mutating at a much faster timescale (milliseconds instead of seconds) and mangling every packet while still enforcing the right OTP for that peer raises substantial technical challenges that must not be discounted.

Crucially, the whole proposal hinges upon the calculated OTP remaining valid when received at the destination endpoint. However, the network jitter becomes very relevant when operating at a milliseconds time frame. Thus, we need either predictable link latency or concede to slower mutations. The Edge's proximity and more predictable latency are instrumental for our mechanism, allowing it to achieve the quicker mutation periods that translate into more effective probabilistic security [115] (*i.e.,* less chance of hitting the target). In turn, Edge computing has far stringent forwarding plane performance requirements due to the low-latency and higher-throughput goals at the root of its inception.

In the following subsections, we will detail the solutions at the core of our design to solve the challenges of peer synchronism, peer management, performance, deployment, and scalability.

4.3.1 Core principles of the TOTP-MTD function

The existing 2FA solutions deployed by the Internet Services already deal with peer synchronization and management. They do so by using the TOTP [110] standard, which relies on the independent calculation of the OTP using the device clock, the shared secret, and the negotiated parameters (such as OTP size and expiration period). Because the synchronization is achieved using the universal time, new peers can be added/removed at any instant without disrupting the synchronization with other peers and their ability to authenticate into the system. No active communication is required between the peers to achieve that synchronization (*i.e.,* no additional overhead, and the new peer is immediately ready to authenticate). Isolated packet-loss incidents do not lead to permanent desynchronization. Furthermore, each OTP has a hard expiration time, limiting an interception attack's usefulness to that validity window.

Our main proposal uses a slightly adapted TOTP function to mutate the service ports, maintaining the familiarity and proofs of a widely known solution. However, using TOTP as the mutation function comes with its own set of challenges. TOTP was created primarily to generate human-readable codes (*i.e.,* decimal values of at least six digits) that last for minutes, making the resulting OTP length unsuitable to the available number of bits that set the listening port in the protocol header. The six digits minimum is also a direct consequence of TOTP's original design having the `SHA-1` hashing function in mind, which did not assure integrity when truncating the digest further than that threshold (*e.g.,* due to the birthday paradox attack). We overcame this obstacle by using cryptographic hashing functions that have indifferentiability in mind (*i.e.,* generating an indistinguishable output from a pseudo-random number generator), notably the `BLAKE2` family [116]. We can now truncate further without changing TOTP's properties or security proofs, other than the evident reduction in calculated OTP's strength (as an authentication key) to the respective bit size. We have kept the dynamic truncation of TOTP [110] using digest sizes compatible with the `SHA-1` and `SHA-2` families used in the design, preserving the original assumptions and proofs. The `BLAKE2` family also allows configuring the

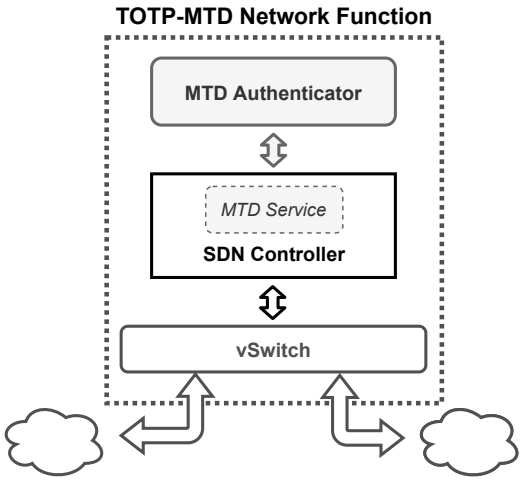


Figure 4.3: Proposed TOTP-MTD NF

hashing function to generate a digest of our targeted bit size. However, using that approach would depart from TOTP's familiarity and proofs, becoming a potential barrier to the adoption.

We have addressed the lowered bit-strength of the OTP by generating new codes more often, going for expiration intervals in the tens of milliseconds instead of minutes. Making the OTPs that much shorter-lived has a substantial impact on the systems' performance and synchronization, vital to making such a mechanism work (at all). The clock source needs to be synchronized up-to the required precision of the mutation, which is achievable with readily available solutions such as Global Positioning System (GPS) time [117], Network Time Protocol (NTP) [118], or Precision Time Protocol (PTP) – IEEE 1588 [119]. We have verified that NTP is sufficient to achieve the required synchronization.

Our solution consists of an SDN application (the MTD Authenticator) that can either work with physical switches or be packaged together with an internal virtual switch inside a VNF (as shown in Figure 4.3). Being a VNF within the ETSI NFV platform crucially allows for the on-demand instantiation and scaling of fully virtualized systems. Crucially, because the IP address is the primary identifier of the peer selection, our horizontal scaling strategy consists of Policy-Based Routing (PBR) to load-balance the peers across the available instances. The PBR approach allows to dynamically redirect the load as the NF scales in/out. Our design also targets the NFV-MANO primitives to provide the required interfaces to add/remove peers across each instance of the solution, fundamental for flexible orchestration and enabling horizontal scalability. Furthermore, the design also allows for forwarding plane hardware offloading to physical network equipment (such as SDN-controlled switches) to make smart use of existing resources during the most computationally demanding phase (*i.e.,* per-packet authentication enforcement). We will further detail the inner-workings in the next subsection.

The TOTP-MTD NF relies heavily on SDN-controlled switches to achieve the stringent forwarding-plane performance requirements of Edge computing, 5G, and Network Slicing. Enforcing the OTP authentication on a per-packet basis would be a very demanding task without a performant datapath, as shown by the prior work. We have specifically chosen an existing set of fields in the TCP/UDP protocol header (the source/destination port) to send the OTPs to minimize data overhead and crucially target this task's offload to an SDN switch. This offloading allows for a more flexible approach than dedicated implementations of a TOTP-MTD datapath: SDN switches are readily available and can be used for other functionalities than just our TOTP-MTD function. However, SDN switches are not designed to calculate cryptographic functions as part of their packet matching rules (*i.e.,* they

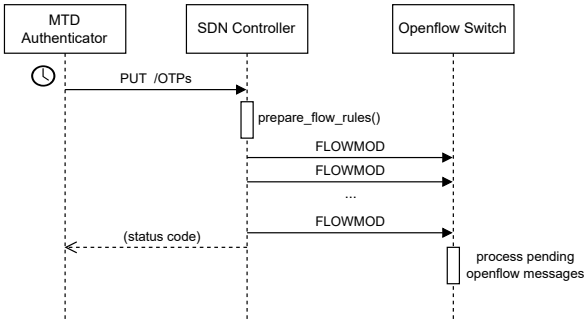


Figure 4.4: Control-plane sequence inside each TOTP-MTD NF

cannot calculate the OTP). The OTP calculation has to be done in the control-plane to make full use of standard equipment via an SDN Application (the MTD Authenticator, as shown in Figure 4.3).

The Authenticator will calculate the valid OTP for each time slot (according to the secret shared among the parties) and post the updated OTPs to the SDN controller at the right time instances (as shown in Figure 4.4). Having the SDN application controlling the timings is a crucial element of the design. It simplifies the controller's logic, allows for different expiration times for different peers, and enables future-work towards a more advanced re-synchronization and delay compensation (*e.g.,* using more data sources). Furthermore, it is easier to integrate the Authenticator with a dedicated secure enclave device (*e.g.,* Hardware Security Module (HSM)).

There is an MTD service built into the controller, serving a dual purpose. The first is minimizing the number of messages in the northbound and their respective sizes compared with issuing requests for many `FLOWMODs`. The second was handling the `PACKET_IN` events when the received OTP does not match, mainly gathering relevant network data (*e.g.,* topology data) and the packet's timestamp before reporting the anomaly to the threat decision function. When packaged together as a VNF (as shown in Figure 4.3), the solution enjoys full on-demand instantiation and scaling. When used alongside a hardware switch, it takes advantage of the efficient hardware offloading that purpose-built resources provide while still retaining the same virtualization benefits to the control-plane functions. The proposed MTD system requires at least two NF instances, one for each end of the connection.

In this solution, the peers are identified by their current endpoint configuration, particularly their IP addresses, as acquired from the Network Access Control (NAC). Both clients and servers can simultaneously connect to multiple endpoints, having different mutations (or no mutation) happening according to the protocol, port, and IP address match. The server shall not expose another service in the same listening IP address and protected protocol, under the penalty of having broken connections once a calculated OTP overlaps with that service's port. Serving additional services using the protected protocol will require another IP address. There are no such caveats for the client.

Upon receiving the peer's OTP list from the Authenticator (Figure 4.4), the controller will then generate flow rules that enforce the valid OTPs (as exemplified in Table 4.1) and issue the required `FLOWMODs` to the switch. Table 4.1 shows the client-side flow table, which is very similar to the server-side except for a *priority=1* rule in *table=0*. That new rule is a trivial safeguard for all non-matching traffic of the protected protocol (*i.e.,* mandatory whitelist operation) that must be reported to the controller through a regular `PACKET_IN`. The Authenticator is then notified of the successful processing of the OTP list into flow rules (*i.e.,* "200 OK"), and the forwarding plane will (eventually) start enforcing the new OTPs sometime after the switch processes the `FLOWMOD` messages. Because time is of the essence in TOTP, the rules generated need to account for packets that fall right at the limit of a time slot, some jitter in the network, or even slight clock skew/drift. We solved this using the

Table 4.1: Sample flow table(s) for a OpenVPN tunnel (client-side)

table=0,	priority=65535,udp,nw_src=x.x.x.x actions=resubmit(,101)
table=0,	priority=65535,udp,nw_dst=x.x.x.x,tp_dst=1194 actions=set_field:3333->udp_dst,output:*(transport)*
table=0,	priority=0 actions=NORMAL
table=101,	priority=65535,udp,nw_src=x.x.x.x,tp_src=2222 actions=set_field:1194->udp_src,output:*(private)*
table=101,	priority=65535,udp,nw_src=x.x.x.x,tp_src=3333 actions=set_field:1194->udp_src,output:*(private)*
table=101,	priority=65535,udp,nw_src=x.x.x.x,tp_src=4444 actions=set_field:1194->udp_src,output:*(private)*
table=101,	priority=0,udp actions=CONTROLLER:65535
table=100,	priority=65535,udp,nw_src=x.x.x.x,tp_src=1111 actions=set_field:1194->udp_src,output:*(private)*
table=100,	priority=65535,udp,nw_src=x.x.x.x,tp_src=2222 actions=set_field:1194->udp_src,output:*(private)*
table=100,	priority=65535,udp,nw_src=x.x.x.x,tp_src=3333 actions=set_field:1194->udp_src,output:*(private)*
table=100,	priority=0,udp actions=CONTROLLER:65535

strategy already specified in the TOTP RFC [110], which considers valid a set number of adjacent OTPs before/after the current time slot (*e.g.,* $[-1, +1]$ valid slots). In the sample flow table (table 4.1), the current OTP for that time slot is 3333. Simultaneously, the accepted receiving OTPs (seen in OpenFlow *table=101*) are three in total, one for the current slot, one for the slot immediately before, and another immediately after. The valid receiving OTPs are updated in two stages to minimize transient states. First, we create a new OpenFlow table with the new OTPs. Then, we update the rule in the default table so that the incoming packets are resubmitted to the new OTPs table (this is a Nicira extension to the OpenFlow protocol). Lastly, we delete the table with the old OTPs, hence clearing those expired codes before reusing the table in a later update.

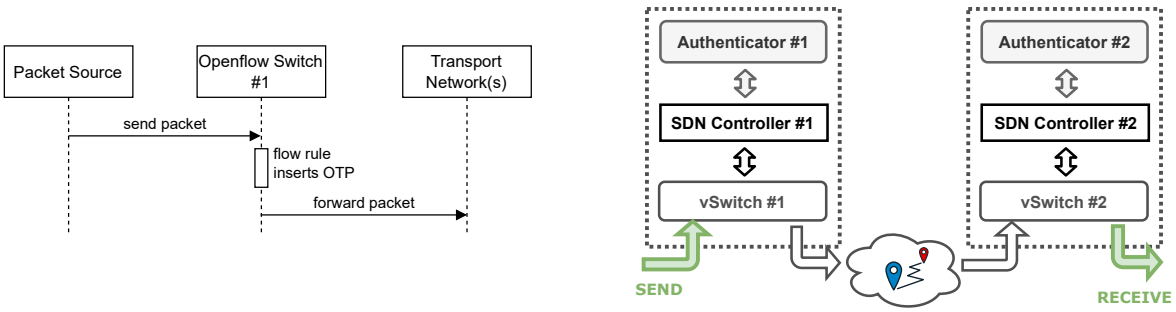


Figure 4.5: Forwarding-plane packet processing sequence for sending side

Meanwhile, in the forwarding plane, the packets are being processed accordingly to the sequence shown in Figure 4.5 and Figure 4.6. The figures illustrate the packet processing sequence taken when sending or receiving a packet between two participating NFs. Each MTD NF's roles are numbered as shown the right of Figure 4.5 (#1 for send, #2 for receive). The same illustration is valid for Figure 4.6, it is just a matter of reversing the roles and numbering. Because we can only have one destination (or source) port in the packet header, inserting the OTP is an unequivocal transformation (Figure 4.5 and the second line of Table 4.1).

The certainty while sending is in stark contrast with receiving the packet (Figure 4.6), which may have multiple valid OTPs accordingly to the set grace period (*e.g.,* $[-1, +1]$ valid OTPs). The same flow rule that matches a correct OTP will also revert the mutation to the protected service's actual port (see OpenFlow *table=100* and *table=101* in Table 4.1). However, when the received OTP does not match, we need to assess if that mismatch is due to a threat or some other innocuous condition (*e.g.,* network jitter). Our solution handles this mismatch through a regular `PACKET_IN` to the SDN controller, which will record the event's timestamp and other relevant network data (*e.g.,* topology) and then send it to a threat decision service. The threat decision can have more data available from

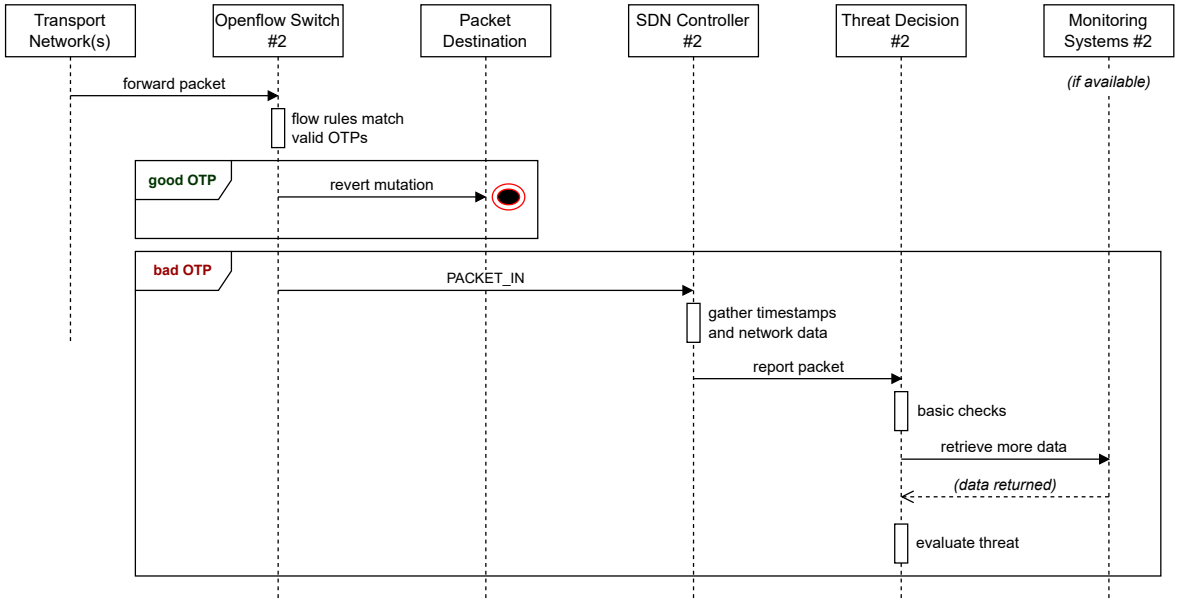


Figure 4.6: Forwarding-plane packet processing sequence for receiving

other monitoring systems of the hosting platform. That will help assess if there was any recorded delay (*e.g.,* network, CPU, or others) that, once factored in, would yield a different received time slot in which that OTP was valid. That will eliminate false-positives for the alarm generation system while still stopping the threats in the data-plane. If the mismatch is not consistent with the regular platform operation, an alarm can be generated so that further analyses are performed to understand the severity of the threat. The balance between eliminating false-positives and not creating too many false-negatives is crucial for our solution because of commonly occurring events such as network jitter.

The smart use of monitoring, well-defined constraints, and the events gathered at the threat detection system would allow the Authenticator to account for the enforcement delays caused by the switch at either end of the communication. It may as well adjust the OTP calculation according to the estimated end-to-end delays so that the OTP remains valid when the packet arrives at the destination (improving performance). As we will show in the next section, even with just coarse monitoring data, the solution is workable and has a suitable performance.

4.4 Evaluation

This section evaluates how our proposal's different requirements and objectives were fulfilled by experimenting with our straightforward PoC implementation in a laboratory cloud environment. We will start by evaluating the hashing function's computational performance and suitability in calculating the OTP. The computational performance is essential for the mutation speed, while suitability is critical for the mutated port's randomness. Then, we will evaluate how the SDN control delays impact our proposal, particularly in keeping both ends of the solution effectively enforcing the same timeslot. The forwarding-plane performance and delays evaluation follow next, being these parameters a make-or-break type of situation for an Edge computing environment. We will then evaluate our solution's threat detection efficacy, focusing on two similar, however separate aspects: (1) how many unauthorized attempts can the solution successfully stop, (2) how many attempts can it report to the human operators. Because the TOTP approach has a natural potential for false-positives due to network jitter (or other commonly occurring events), we will also explain the practical trade-offs

introduced when assessing if an event is a false-positive. Lastly, we will experiment and critically evaluate how the proposed TOTP-MTD NF scales.

4.4.1 Hashing performance and suitability

Computation time matters for our solution, as the timeslot is keyed in each of the quick mutations. The defensive abilities of the proposal rely upon changing the calculated OTP often. Because the TOTP function employed is a standardized way of using an Hash-based Message Authentication Code (HMAC), it is essential to determine how each hashing function performs while computing the TOTP. The performance of a hashing function depends not only on its algorithm but also on the amount of data it needs to process. All of the evaluated hash functions have a set block size that varies from algorithm to algorithm. Therefore, the number of operations required to process a given size secret may also vary depending on the selected function. We have conducted this evaluation focusing on the practicality of its users. Instead of calculating the optimal secret sizes for each particular function, we have evaluated the performance when using the standard key sizes of most generators (160, 512, 2048, 4096, 8192, 16384 bit).

It is critical to note that, due to the way TOTP works and because of our particular OTP size requirements, the chosen hash function must allow the truncation to 16-bit values. This restriction immediately excludes the `SHA-1` family, which is at the core of the original TOTP design. Fortunately, TOTP was built with replacing the hashing function used in the HMAC calculations in mind, allowing for a specific set of other hashing functions with a compatible digest. We have exploited this characteristic to build our proposed mechanism, introducing more modern cryptographic hashing functions with a compatible digest (such as the `BLAKE2` family).

We present in Table 4.2 the computation time (in *microseconds*) that each hashing function takes to calculate a TOTP with that given secret key size. The secret key was randomly generated at each calculation to avoid any potential biases caused by caching. For each hashing function, we have conducted $2^{16} \times 10^3$ experiments to have statistically significant results. To improve the reading of such a large number of values, we have presented each hash function's performance relative to the best result within that key size (in percentage), being that baseline shown in its original notation.

Table 4.2: TOTP calculation time using different hashing functions (varying the secret key size) – unit is microsecond (μs)

Hashing Function	**Key Size**						
	160-bit	**512-bit**	**1024-bit**	**2048-bit**	**4096-bit**	**8192-bit**	**16384-bit**
blake2s	**8.86 ± 3.70**	**8.39 ± 2.95**	+18.53%	**10.74 ± 4.15**	**11.85 ± 4.50**	+4.13%	+10.30%
blake2b	+1.68%	+9.04%	**8.63 ± 2.53**	+2.23%	+0.34%	**13.43 ± 5.18**	**15.82 ± 5.45**
sha224	+24.84%	+27.87%	+55.59%	+23.44%	+22.69%	+21.42%	+31.22%
sha256	+26.98%	+27.83%	+55.20%	+25.31%	+22.69%	+21.40%	+31.36%
sha384	+41.21%	+38.92%	+33.91%	+34.38%	+29.78%	+22.47%	+22.04%
sha512	+40.21%	+38.68%	+34.14%	+34.22%	+28.17%	+22.64%	+22.06%
sha512-224	+36.67%	+40.13%	+34.22%	+33.39%	+27.09%	+21.50%	+22.11%
sha512-256	+37.93%	+39.61%	+34.93%	+34.67%	+28.83%	+22.49%	+23.22%
sha3-224	+59.41%	+68.50%	+59.78%	+59.75%	+54.84%	+54.82%	+64.83%
sha3-256	+59.52%	+64.41%	+58.74%	+58.96%	+53.34%	+55.31%	+69.13%
sha3-384	+61.66%	+62.96%	+92.61%	+62.85%	+57.47%	+69.54%	+82.13%
sha3-512	+60.07%	+63.84%	+93.22%	+70.18%	+71.83%	+84.65%	+117.20%
sm3	+39.61%	+37.20%	+70.14%	+42.85%	+43.79%	+53.08%	+81.08%
whirlpool	+62.08%	+55.87%	+89.78%	+65.16%	+69.46%	+93.15%	+130.51%

Table 4.3: TOTP randomness quality of the generated network ports (per hashing algorithm)

Generator Function	entropy	Δ to mean port	π	gzip	lz2	lzma
blake2s	15.999 ± 0.000 bits	−0.66	+0.13%	+0.03%	+0.44%	+0.01%
blake2b	∼	−1.62	−0.18%	∼	∼	∼
python3 secrets	15.999 ± 0.000 bits	+0.58	+0.06%	+0.03%	+0.44%	+0.01%
null hypothesis	15.999 ± 0.000 bits	−0.27	−0.12%	−13.91%	−49.14%	−45.92%

The results show that the `BLAKE2` family is the fastest hash function for our purposes, with the family of `SHA-2` hash functions following next ($\approx$ 25% *to* 40% slower than `BLAKE2`), and the `SHA-3` family performed even worse ($\approx$ 55% to 120% slower). Less commonly used functions such as `sm3` or `whirlpool` were not competitive either.

The truncation of the HMAC should not change its pseudo-randomness output quality, assuming the use of a hashing function with indifferentiability in mind (*i.e.,* generating an indistinguishable output from a pseudo-random number generator), such as the `BLAKE2` family. We have experimentally verified if there was any evident adverse effect caused by the 16-bits truncation of the HMAC instead of the initial six decimals limit, which arises from the TOTP's original design focusing on the `SHA-1` family. As presented in Table 4.3, we have compared the results of the TOTP calculated using the `BLAKE2` family against a cryptographically secure pseudo-random generator (the *secrets* module of Python3). We have also tested the null-hypothesis using a custom-built faulty pseudo-random generator with 16-bit output. Only eight bits are pseudo-random, being the other eight carried-over over from the previous value (hence the fault). Properly evaluating randomness quality is a very complex task. We have opted to first verify some of the necessary conditions of a proper random sequence, much like Fourmilab's ENT test[1], albeit checking necessary conditions may not be sufficient to detect the null hypothesis (as we show in Table 4.3). The tests we used to check for necessary conditions include the resulting entropy of the generated set, the deviation of the average generated value to the ideal mean, and the pseudo-random outputs of the OTP to calculate the value of π using the Monte Carlo approximation method. All functions passed the necessary conditions tests, even the null hypothesis, highlighting the need for carefulness when trying to extrapolate certainty metrics that characterize uncertainty (*i.e.,* randomness). We then successfully used the Lempel-Ziv test [120] to detect the null hypothesis, which consists of attempting to compress the data to detect patterns in the set of randomly generated values. Unlike the known good generator (Python3 *secrets* module) and the `BLAKE2` hashing functions that were incompressible, the null hypothesis was compressed up-to $\approx 50\%$, which was a very successful detection of the pattern of the faulty generator.

After completing both the performance and quality tests, we have selected the `BLAKE2b` hashing function to conduct our PoC. This function allows for the best compromise between key size and calculation time (having a 1024-bit key, the OTP was calculated within 8.63 ± 2.54 μs). The quality tests did not indicate any issue with the approach taken.

4.4.2 SDN control overhead and delays

This subsection will measure the delays caused by the SDN control overhead and message queue processing at the switch. To do so, we have slightly modified the MTD service inside the SDN controller to send a `BARRIER REQUEST` after sending all the necessary flow mods. The `BARRIER` messages are a mechanism built into OpenFlow to signal that the previously sent commands were already taken

[1] https://www.fourmilab.ch/random/

from the switch queue for processing. We recorded the timestamps of sending the barrier request and receiving the barrier reply (as shown in Figure 4.7). Because determining a command was taken for processing is quite different from that command being already enforced, we have also created a purpose-built service to measure the full end-to-end delay in the forwarding plane (as shown in Figure 4.8). That measurement service is a network function that listens on two sockets (the current and the previous OTP) and records the time delta that goes between receiving a packet in one socket (previous OTP) and the next packet in the other socket (next OTP). To make a reliable measurement, we have fixed the listening port numbers and alternated amongst those when issuing the valid OTP for that time slot. We also had a UDP client sending a packet with a reasonable precision timestamp (up to the microsecond) every 0.1 ms.

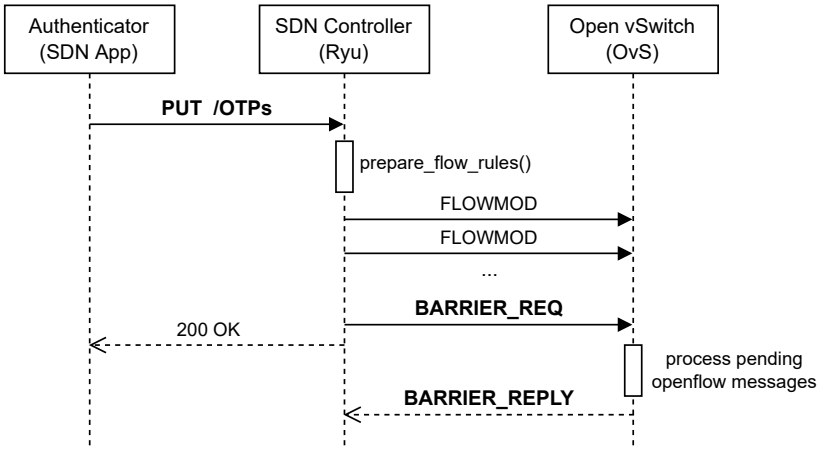


Figure 4.7: Sequence diagram of the SDN overhead evaluation

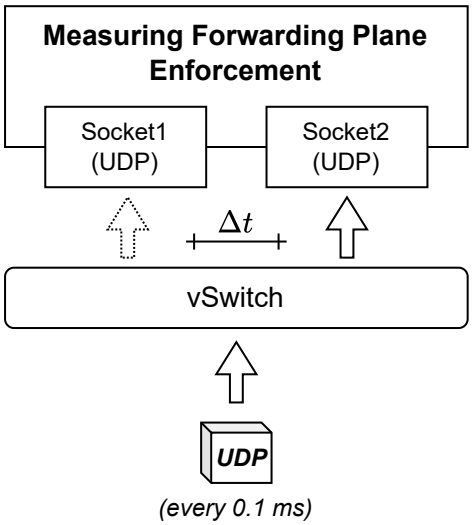


Figure 4.8: Network Function created to measure the enforcement delay

The full breakdown of the measured delays after 100 000 experiments is presented in Figure 4.9. In the control-plane, approximately 45% of the average total time (5.44 ms) was spent sending the OTP list via the northbound REST interface. The MTD SDN service built into the controller then spent little time ($< 2\%$) generating the required flow tables, issuing the respective `FLOWMOD` commands to the switch and then finishing with the barrier request message. The switch then took about 26% of the total time to process the pending OpenFlow messages and generate the barrier reply. However,

the flow rules are only enforced a few moments after the switch processes the messages. The total end-to-end enforcement delay measured in the forwarding plane was $5.44 \pm 1.13\ ms$.

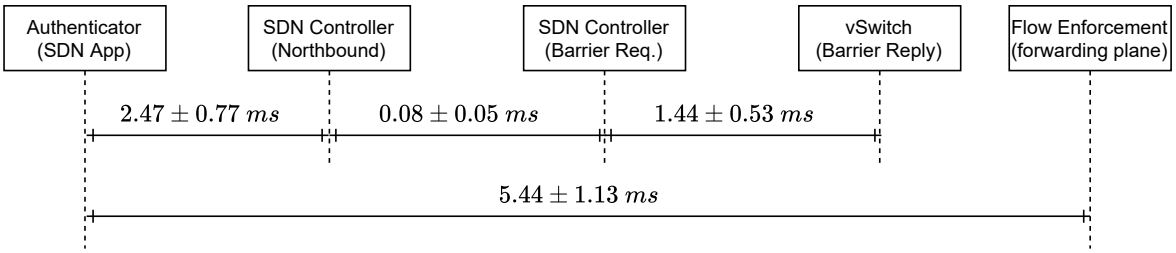


Figure 4.9: A breakdown of the measured SDN control delays of the solution

For our purposes, the $\approx 5\ ms$ figure is not worrisome or an immediate obstacle in itself. Because the Authenticator (SDN application) controls the clock used for the OTP calculation and validation, any accurately predictable delay can be compensated (*i.e.,* we need to offset the calculated time slot by the right amount of delay). However, having the standard deviation amounting to $\approx \pm 20\%$ of the $\approx 5\ ms$ enforcement delay will constrain the more straightforward implementations to mutation periods in which that deviation is not significant ($\pm 1.13\ ms$). More advanced AI/ML solutions that leverage monitoring data to make better predictions and adjustable delay compensation according to control-plane load may enhance the faster mutation periods' reliability.

Furthermore, unlike our PoC that uses the same implementation across very similar hardware for both endpoints, it is critical to account for the different processing delays when there are differently performing endpoints trying to communicate with each other. Otherwise, failure to set the correct end-to-end delay of the other endpoint will cause both endpoints to become out-of-sync, and the solution will not work correctly.

We must note that the way this solution is designed, the SDN control does not impede any existing forwarding plane rules. Therefore, the SDN control delays do not translate to a similar magnitude of delay in the forwarding plane. We will evaluate the behaviors of the forwarding plane in the next subsection.

4.4.3 Forwarding plane overhead and delays

The impact of forwarding plane performance is one of Edge's most critical aspects (and 5G) because network latency and throughput are among the most commonly used KPIs for end-application requirement specification. Therefore, in this subsection, we will evaluate the forwarding plane network performance of our solution. To exclude external variables' contribution in our results, such as Network Interface Controller (NIC) or cabling limitations, we will use a single large node (VM) that containerizes the different networks using Linux Network Namespaces. Each of the namespaces will be terminated in a separate Open vSwitch bridge with its separate SDN controller instance. In effect, our network throughput is now limited just by the compute node's memory bandwidth and CPU power, revealing any performance bottlenecks that would otherwise not be noticeable when using NICs. Because our solution relies on the synchronization of clocks between the sending and receiving endpoints, we cannot evaluate that factor correctly when using a single VM (*i.e.,* both ends share the same clock). Therefore, we have also repeated the experiments using two separate VMs hosted in different compute nodes. The clock synchronization between the nodes was achieved using NTP. We have validated that the solution behaves similarly to the single VM evaluation. As expected, the two VMs experiment was constrained by the NICs throughput and had added latency due to cable distance and network stack

overheads to interact with the NIC. The added latency and jitter of that interconnecting network also influenced the mutation period's absolute fastest boundaries, but the findings remained similar. We must stress that, when independently reproducing the results, the Authenticator (SDN app) must be adjusted with the proper delay offsets caused by link latency and different computational capabilities that impact control-plane operations. In our case, both compute nodes had similar capabilities (same make and model, with Intel Xeon E5-2620 v4 CPUs) and were running the exact same implementation of the MTD mechanism. Therefore, we only had to account for an average symmetric link latency of ≈ 0.46 ms (*i.e.,* the offset used for OTP calculation).

We have measured link latency using a custom UDP client and server that measures the one-way link latency through a timestamp (up to the microsecond) embedded into the packet payload. A new packet is generated every 0.1 ms. We have repeated $25000 \times mutation\_interval$ runs for each mutation period to have statistically significant data at the timeslot change. Figure 4.10 and Figure 4.11 shows that the measured link latency closely resembles the baseline latency (*i.e.,* without mutations) up until the mutation period exceeds 15 ms. After that point, there is a slight increase whose total measured latency is < 0.1 ms on average. However, there was also a small increase in jitter, starting at 10 ms. Once the mutation period exceeded the 7 ms, the system started having dropped packets, which were sent correctly to the controller and recorded in the threat assessment system as missed OTPs. The 7 ms threshold is compatible with the SDN control overheads measured in the previous section. The 7 ms mutation period is the inflection point at which the deviation of the control-plane delays (5.44 ± 1.13 *ms*) must have started to become significant. After crossing this threshold, the mutation periods are no longer enforced correctly but rather as fast as the system can change the OTPs (as verified by traffic logs). Because our PoC runs the same implementation on both ends within the same VM, the solution may keep working beyond the control-plane delays. The systematic errors will be very similar on both ends, achieving synchronization despite not enforcing that mutation period correctly. Although no longer keeping up with the mutation speed in these particular circumstances of a single node, the number of packet drops still stays below 1% until 3 ms. After this point it rises dramatically, starting at 2.25% in 3 ms, then 6.26% in 2 ms, and finally 14.40% in 1 ms. The experiments conducted using two nodes revealed that our straightforward implementation behaves similarly in UDP latency to the depiction in Figure 4.10 and Figure 4.11 up-to-the identified threshold of the SDN control limits. After that, we get a more substantially rising packet loss, similar to the TCP throughput test's behavior (Figure 4.12 and Figure 4.13). We have opted not to include separate plots for the internode tests, as they are similar to the single-node test.

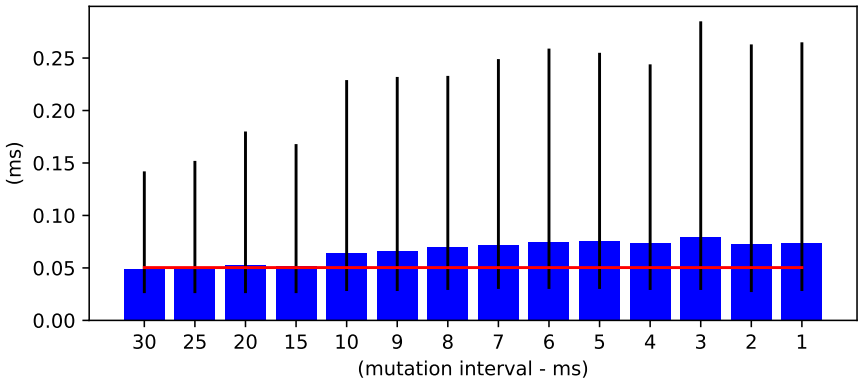


Figure 4.10: UDP Latency (typical)

We have used *iperf* in TCP mode to measure the throughput of our solution. The client only

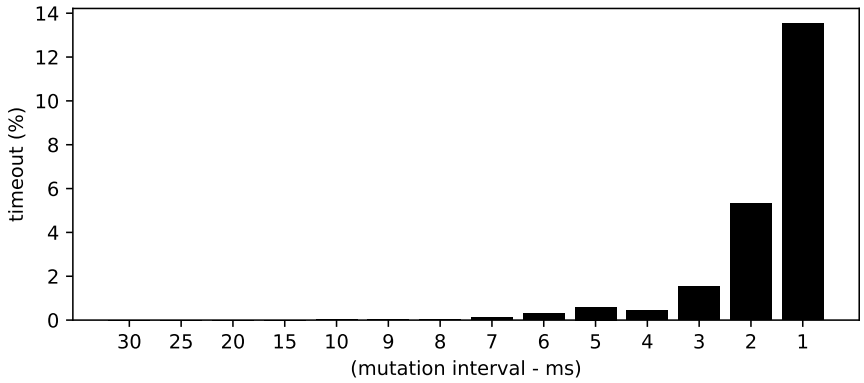


Figure 4.11: Timeouts during each UDP Latency run (average)

uses one connection, and each test usually completes in about 10 seconds. We have performed at least 250 runs for each mutation period and set a timeout of 30 seconds for each test. We have also recorded the CPU usage during the test, focusing mainly on our virtual switch (that must mangle such a high number of packets) and the SDN controller (that must handle the `PACKET_INs` from mismatched OTPs). The CPU load was measured using Python's psutil[2] and a sampling interval of 100 ms to make the CPU bursts evident in otherwise lower averaged loads. Due to constraints imposed by the test setup, the recorded Open vSwitch CPU load pertains to both ends of the MTD connection (*i.e.,* both sender and receiver). The SDN controller load refers to just one of the ends, and we have verified that both client and server had similar loads. The baseline refers to a pro-active SDN control approach in which the vSwitch gets a single catch-all rule that does the action `NORMAL` (*i.e.,* a plain switch with no significant interaction with the controller).

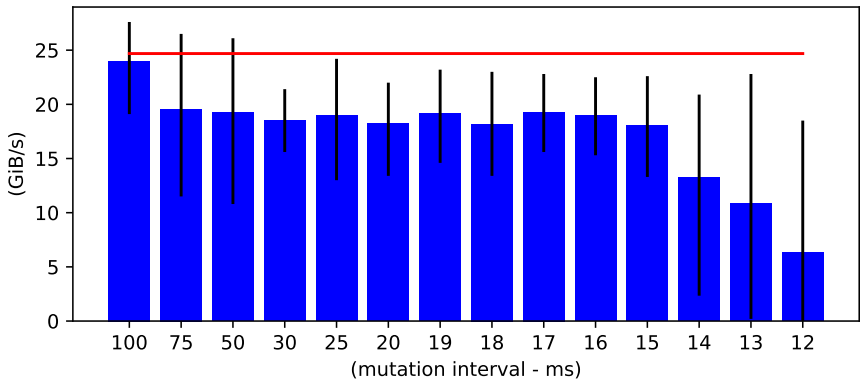


Figure 4.12: TCP Throughput

The throughput is very close to the average baseline when the mutation period is 100 ms ($\approx$ 24.0 GiB/s vs. 24.65 GiB/s), falling within the error margin. There were no connection timeouts or lost packets due to bad OTPs in the 100 ms runs. However, the CPU load on both the vSwitch and the SDN controller was substantially higher than the baseline due to the frequent flow updates and higher demands of packet mangling versus plain pro-active forwarding. In the 75 ms to 15 ms mutation periods, the throughput lowers to $\approx$ 18 to 19 GiB/s. The CPU load starts steeply rising as we change the packet mangling rules faster (see the OvS CPU load in Figure 4.14). While the throughput was quite good from 75 ms to 15 ms, and there were no connection timeouts during the test, we did observe

[2]https://psutil.readthedocs.io/en/latest/

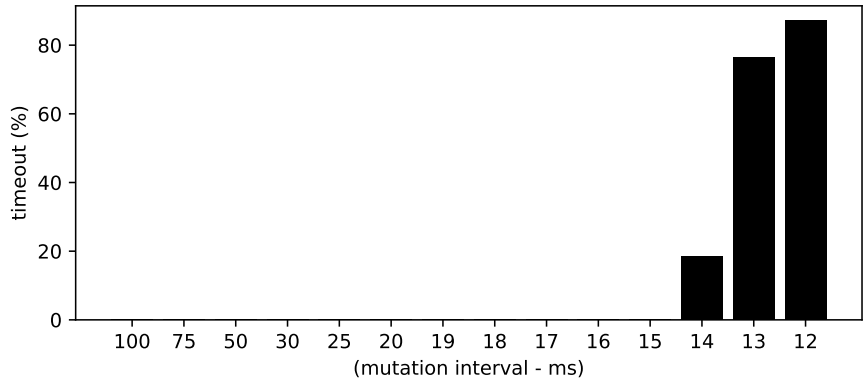


Figure 4.13: Timeouts during each TCP Latency run (average)

a rising number of lost packets due to missing the right OTP for that timeslot. Those lost packets were successfully recovered by the TCP session and are likely the throughput loss cause compared to the baseline. We will go more in-depth into these lost packets in the next subsection. The increasing number of `PACKET_IN` events from those missed OTPs is also the likely cause for the rising CPU load in the SDN controller. The number of connection timeouts rises steeply after the 14 ms mutation period, and the solution becomes too unreliable after the 13 ms mutation period (over 75% connection timeouts). The CPU load dramatically falls as we start having timed-out connections. This reduction is because we have far fewer packets in the network while the TCP connection tries to recover. We must note that the throughput numbers only consider the runs in which a numerical value was acquired (*i.e.,* without a timeout). Therefore, the throughput numbers for the mutation periods 14 ms to 12 ms must not be used to infer the total amount of data that went through the system in those runs.

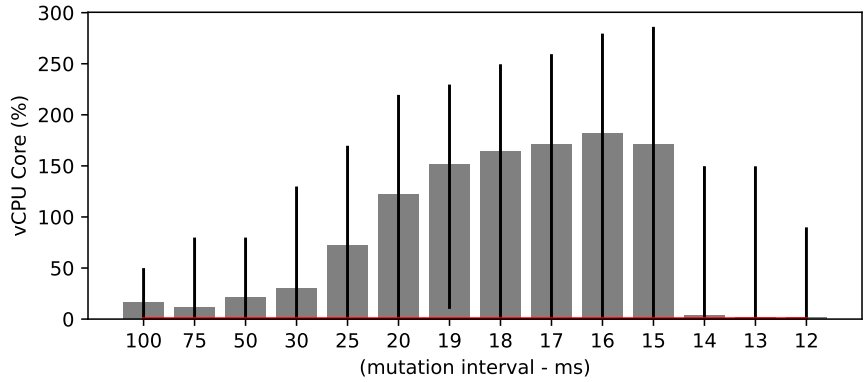


Figure 4.14: Open vSwitch CPU Load (combined)

Despite the straightforward implementation of the PoC, which did not feature any adaptive compensation for jitter in the system, the experimental results show that this proposal can have little impact on the forwarding plane performance as long as the mutation period is correctly tuned. However, we do have non-negligible computational requirements for both the SDN controller and the virtual switch, which can rise very steeply as the mutation period accelerates. Nevertheless, because the vSwitch causes the most considerable CPU overhead and our solution employs SDN control, it may be possible to offload that CPU usage to a hardware OpenFlow switch.

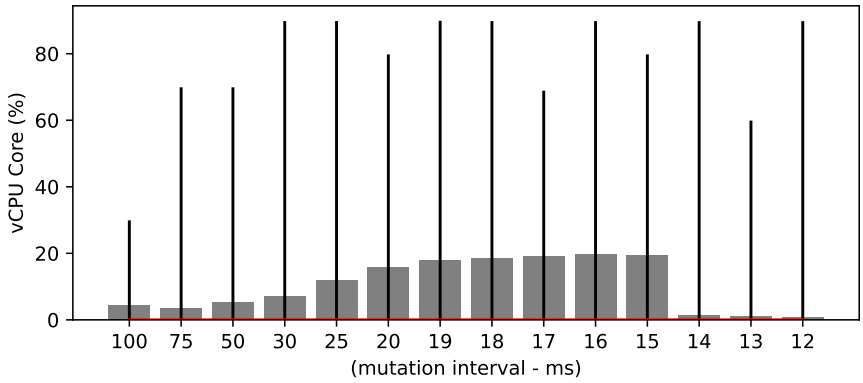


Figure 4.15: SDN Controller CPU load

4.4.4 Threat detection efficacy

Aiding in threat detection is one of our solution's main contributions, alongside making service enumeration and exploitation harder (*i.e.,* stop unauthorized probing) to the attacker that does not hold the movement's secret. Because network jitter and other variances may cause an OTP to miss its targeted time slot, we need to evaluate our solution's effectiveness in separating these regularly occurring events from serious threats (*i.e.,* identify the attackers trying to enumerate services). As the highly stressful TCP throughput tests have already shown, without considering the massive number of packets transacted in the network, the raw number of flagged events for further threat analysis even without adversarial action was substantial ($\approx$ 841 thousand events throughout those tests). We will start by looking into the experimentally more controllable UDP traffic to analyze those OTP misses and determine why those misses happened despite having no adversarial action. In the next subsection, a method to eliminate the false-positives will be introduced. In the last subsection, we will reassess the capabilities to stop and detect adversarial actions in light of the false-positives elimination.

4.4.4.1 Eliminating False Positives

We found a noticeable latency increase in the forwarding plane performance section when going from 15 ms to 10 ms. After repeating the experiment, and taking a more in-depth look into the missed OTPs in that threshold's vicinity. The results show there are significant differences between the before and after that threshold (Figure 4.16). Before crossing the 15 ms mark, there are very few OTP misses – only four were recorded. It could be determined the missed timeslot in three of them (shown in Figure 4.16), but the other miss was completely extemporaneous. While there are very few misses in the $[30, 15]$ ms mutation interval, a miss will be due to a considerable delay (more than 100 timeslots).

In contrast, when the mutation period yields an increased forwarding-plane latency over the baseline ($[14, 10]$ ms), there will be more missed OTPs. However, the vast majority is just a few timeslots apart (averaging nearly two slots). This finding is crucial to eliminate the false positives, especially when no additional monitoring data is available from the platform, such as in our straightforward PoC.

The solution allows setting different grace periods (*i.e.,* valid OTPs) for stopping suspicious behavior (*i.e.,* forwarding plane rules) than when alerting suspicious behavior (*i.e.,* threat decision alarm generation). That is, it can reduce the number of alarms that need more in-depth inspection while still enforcing stricter tolerances on the forwarding plane by merely setting different grace periods in the respective components. The configuration space has 65535 values possible (the number of service ports). For example, suppose our threat decision system validates the nearing $[-655, +655]$ timeslots

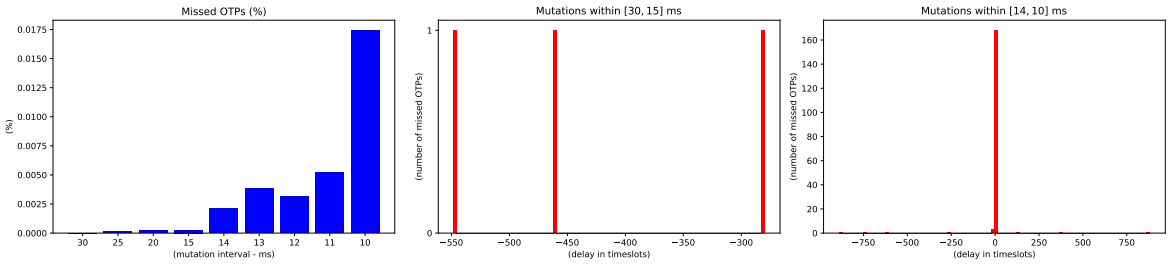


Figure 4.16: Bad OTPs reported to the threat assessment system (without adversarial action)

before generating an alarm. In that case, it will cover all events shown in Figure 4.16 while still only having a $\approx 2\%$ theoretical probability of creating a false-negative. There is just one false positive that survives this elimination process, in over 7.5 million messages considered for the experiment. The threat assessment system could quickly eliminate such a false positive using additional anti-malware functions over the suspected packet/flow. In the worst case, the system administrators should handle a single false alarm easily.

In sum, the system's security relies on an adjustable trade-off between false-negatives and false-positives, which is dictated by the available resources, additional anti-malware functions, and monitoring data. It is shown that this solution allows us to eliminate almost all false reports generated when there is no adversarial action. The discussion will now move to the real goal, demonstrating that it can detect adversarial action (probing attempts) against the network.

4.4.4.2 Stopping and Flagging Adversary Action

It is considered that an attacker is placed into the protected network, which somehow already identified the NF address that holds the sensitive NFV-NS (*e.g.,* through the same configuration leakage that allowed him into that network). The attacker generates a single probing attempt, at a random time, in a bid to identify the listening port of that sensitive service while remaining undetected. It was measured the number of times the attacker avoids detection through a Monte Carlo experiment. The attacker uses the best method to guess the service port, a randomly generated port. The experiment was repeated 100 million times to get statistically significant data.

The threat detection happens in two stages. The first is in the forwarding plane, where the rules' enforcement only allows a tolerance of $[-1, +1]$ timeslots, blocking and reporting any mismatching packets to the threat assessment system. The second stage is after the blocking event in the forwarding plane; the threat detection system will assess that event and determine if an alarm needs to be generated or if that report is due to regularly occurring conditions (*e.g.,* CPU load or network jitter). The grace period is the same devised in the previous subsection to filter the false positives, $[-655, +655]$ timeslots.

The attacker may avoid detection in two different situations. The first and most severe, in which the attacker matches one of the valid OTPs in the forwarding plane and hits the real service, has a measured probability of $0.0045 \pm 0.0005\%$. The second and less severe one is when the probing attempt was stopped in the forwarding plane (*i.e.,* no useful outcome to the attacker). However, the threat assessment system produced a false-negative from that reported packet, therefore not generating an alarm. The probability of having a false negative was measured to be $1.9768 \pm 0.0099\%$. Therefore, when both cases are considered together, the measured probability of that single probe not generating an alarm is $1.9813 \pm 0.0099\%$. It must be noted that if the attacker attempts to do multiple probes, then the chances of detection improve substantially, which curbs the asymmetric relationship the

attackers had over defenders.

The results demonstrate the proposal's effectiveness, as an extra layer of security, in stopping attacks against exposed network services and the soundness of the options made from the previous subsection's experimental data to remove the false-positives. Furthermore, the PoC in the evaluation was a straightforward approach that does not leverage newer technologies (such as AI/ML) to improve delay prediction. The PoC also did not have access to live monitoring data accurately depicting the network's and compute nodes' statuses. Despite that, the experimental results already show that the proposed TOTP MTD is sound for Edge/5G/Network Slicing systems.

4.4.5 Scalability

In the course of the evaluation, it has been experimentally shown that the TOTP-MTD approach works and is effective in different conditions for at least one client communicating with the sensitive service. Now, it is relevant to evaluate how the solution scales when multiple clients communicate with the sensitive network service. This subsection will start by analyzing the function's horizontal scalability design and determining the absolute maximum peers per single instance. Then, an experimental evaluation of how each instance of the TOTP-MTD NF behaves when faced with a varying load was performed, starting with one peer up to the absolute maximum.

The TOTP-MTD NF approach allows scaling horizontally by design, distributing the load of the sensitive service's authenticating clients across multiple instances. Because the identifier that selects an authenticating peer relies firstly on its IP address, the horizontal scaling can be achieved using PBR to select the right instance to communicate with the client IP. It has been experimentally validated that such a scaling approach is feasible. Horizontally scaling this proposal has the same requirements as scaling any other VNF. The scaling process requires additional orchestration and monitoring to assure proper load-balancing (*e.g.,* that enough CPU is available to not overrun the authenticating timeslots) and efficient use of resources (*e.g.,* that we do not overprovision the resources allocated to these functions). Because of this horizontal scaling approach, the solution should only be constrained by the resources available to run the required number of instances that would cope with that load.

Thus, it must be crucially assessed the load behavior inside each instance before starting to scale horizontally. The TOTP-MTD function relies heavily on OpenFlow table rules to perform/validate the mutations. That was a deliberate choice to minimize transient states when changing OTPs: redirecting to a new table is, at its core, a change of a byte in the main flow table. A single byte can only resolve $2^8 = 256$ tables, and some of those tables are not usable (table zero was reserved for the redirect, table 254 was already taken by the switch, and table 255 dumps all flows from the switch). The IP address selects each authenticating device. Because the solution is switching between tables that hold the valid OTPs for that peer, two tables per IP address are needed (*i.e.,* the currently enforcing OTPs and the newly introduced OTPs). Therefore, in its present form, each TOTP-MTD NF instance cannot authenticate more than:

$$maximum\_IPs\_per\_switch := \left\lfloor \frac{usable\_openflow\_tables - reserved\_tables}{tables\_required\_per\_IP} \right\rfloor = \left\lfloor \frac{(2^8 - 2) - 1}{2} \right\rfloor = 126$$

It was experimentally validated the TOTP-MTD NF instance's scaling within its limits of $[1, 126]$ IP addresses, each with a different authenticating key. Although the PoC implementation was never meant to authenticate many concurring peers, as the OTP calculation is single-threaded and sub-optimal

for multiple peers, the results still show the solution works as expected regardless of operating at its upper limit.

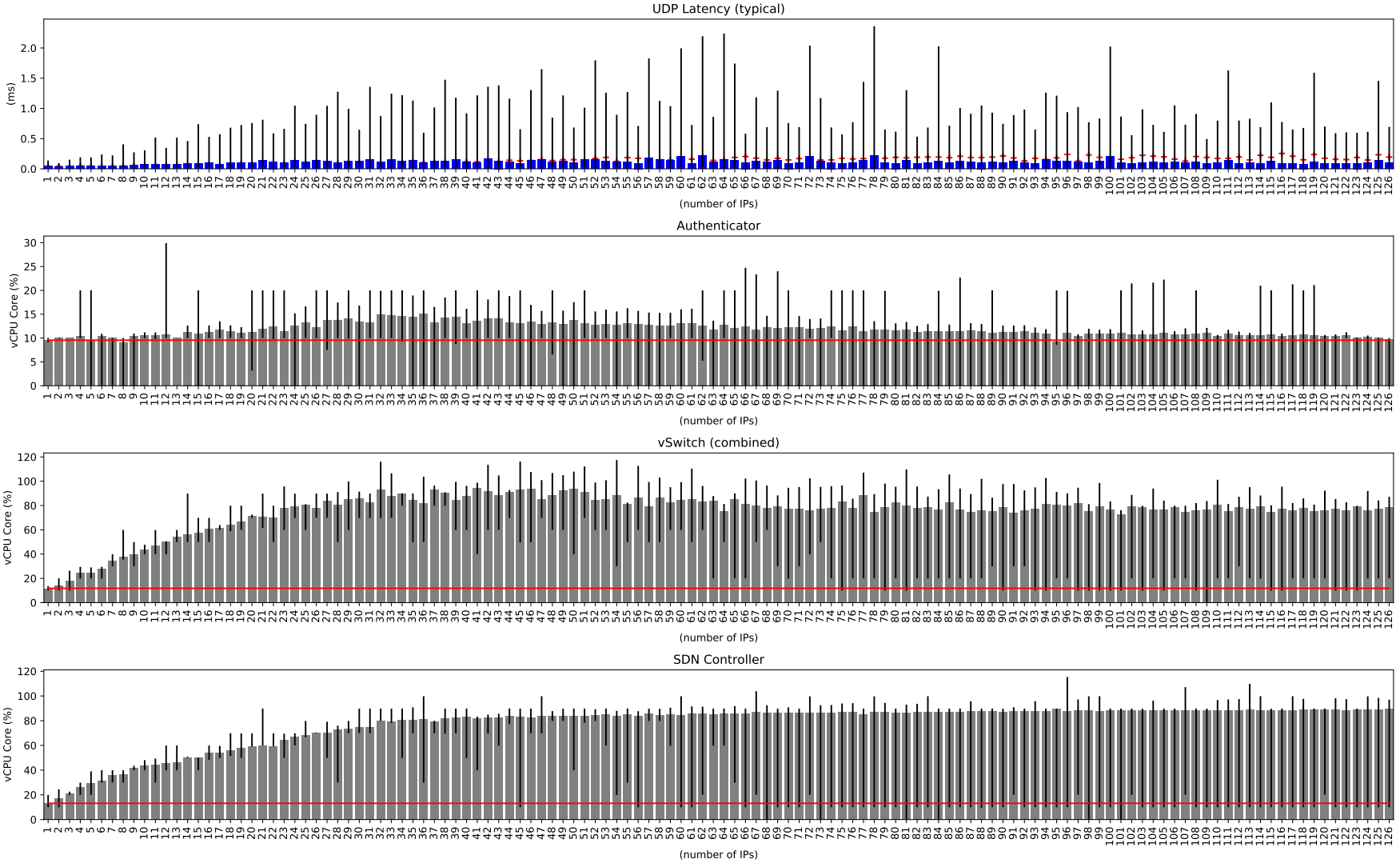


Figure 4.17: Analyzing the TOTP-MTD scalability exploring the full-range of possible peers (per each NF instance)

The results from the scalability tests are depicted in Figure 4.17. Starting with the latency in the forwarding-plane, it could not be observed any substantial deviation that was not already present in the control (*i.e.,* without the MTD system). The latency measured at scale tracks correctly with the earlier experiments that had a single device. Similarly, the Authenticator does not consume substantially more CPU because of the added devices. That is because of two reasons. First, the code is single-threaded; therefore, it could not voraciously use all CPU cores to calculate as many OTPs as fast as possible. Secondly, it was already determined that the OTP calculation was not the most demanding operation in the previous sections. It was already observed that the SDN vSwitch and the respective SDN Controller were the most demanding components of the solution, increasing the demands as the mutation period shortened. In Figure 4.17, it is shown that the same behavior also happens when increasing the number of authenticating peers. In this experiment, was chosen a mutation time (30 ms) that yield approximately the same starting CPU load ($\approx 10\%$) across the Authenticator, vSwitch, and SDN Controller when authenticating a single peer. It was also verified the same behavior for different mutations but have not determined the new maximum mutation speed when under heavy parallel load. That benchmark yields hardware-dependent results with lower value to the scalability evaluation.

The proposed TOTP-MTD solution scales well. The Authenticator does not present a substantial load increase when varying the number of peers per instance. The most CPU-consuming components (vSwitch and SDN Controller) have their loads rise up-to an early plateau and remain workable throughout the entire operating range.

4.5 Summary

Our TOTP-MTD solution successfully delivers a suitable port-mutation mechanism for deployment in Edge, 5G, and other Telco systems. The design and technical approach solved the decisive challenges hindering the existing port-mutation solutions: maintaining mutation synchronism while delivering the necessary flexibility and soundness.

TOTP eliminated the need for an exclusive peer synchronization protocol, avoiding those overheads and other solution scalability issues. Our experimental results demonstrate that the mechanism works with excellent forwarding-plane performance and can tolerate packet loss without losing synchronization permanently. Crucially, the mechanism can sustain quick mutations (in the order of milliseconds) even with multiple peers and unique keys per peer.

Having NFV at the core of its design and packaging the solution as a VNF allowed great flexibility. We could leverage the NFV-MANO interfaces for necessary actions such as adding/removing peers, configuring mutation parameters per-peer, and crucially enable horizontal scalability. We have successfully validated the horizontal scalability using PBR and shown the effects of adding peers to a running instance (within its full operating range). Forwarding-plane performance and OTP generation load remained mostly unaltered while the enforcement's CPU load rose up-to a plateau. Furthermore, our design allows offloading the demanding enforcement to regular SDN switches.

TOTP-MTD reuses the proofs and protocols of an already vetted 2FA solution, not a custom function that may require recertification for industrial compliance. The time-based approach puts a hard limit to the window of opportunity in traffic interception/MitM attacks. We have experimentally validated that the approach stops unauthorized access and detects threats to the system (*e.g.,* probing attempts). The solution design sets distinct thresholds for enforcement and alarming. We enforced stricter limits in the forwarding plane (to stop adversarial action) and then devised a false-positives

elimination method in the alarming phase.

We have achieved excellent all-around performance (latency, throughput, OTP enforcement, and threat detection) despite our PoC implementation's straightforwardness. However, there is still room for improvement (future work). The use of active monitoring and other monitoring data gathered by the Edge computing platform (as in 5G vertical systems) could significantly reduce the threat assessment system's false-negative rate. The grace periods could be adjusted according to the actual measured conditions that affected that particular packet. Furthermore, AI/ML to perform delay prediction and adaptive adjustment could significantly reduce the forwarding plane's false-positive rate, possibly unlocking quicker mutation periods.

In conclusion, our TOTP-MTD approach delivers crucial improvements over the existing port-mutation mechanisms that make the approach suitable for deployment in softwarized and virtualized. The improvements fall under three decisive areas, which we will summarize in the following paragraphs.

Significant improvements in the mutation synchronism: *(i)* We do not need an exclusive peer synchronization protocol that consumes additional bandwidth and hinders scalability at quick mutation rates. *(ii)* The peers synchronize independently, without requiring any active communication between each peer. *(iii)* The mechanism tolerates packet loss without losing synchronization. *(iv)* Crucially, the mechanism can sustain quick mutations (in the order of milliseconds) even with multiple peers and unique keys per peer.

Significant improvements in flexibility: *(i)* The approach allows adding/removing peers at any time. The key enrollment retains the familiarity and advantages of existing 2FA systems. *(ii)* The NFV+SDN design allows for horizontal scalability (i.e., distribute load across instances) and hardware offloading (i.e., the per-packet authentication is delegated to regular SDN switches). *(iii)* The mutation parameters, key enrollment, and scaling are possible to orchestrate by an NFV-MANO solution.

The mechanism's soundness: *(i)* The cryptographic fundamentals reuse the proofs and protocols of already vetted 2FA solutions, not custom functions that may require recertification for industrial compliance. *(ii)* We can attain exceptional forwarding-plane performance despite the very significant packet mangling caused by the mutation. *(iii)* The time-based approach puts a hard limit to the window of opportunity in traffic interception/MitM attacks. *(iv)* The approach is effective at stopping unauthorized access and at detecting threats into the system.

CHAPTER 5

Enabling Enhanced Response and Information Gathering

This chapter introduces a new smooth handover mechanism to enable enhanced response to an on-going attack and perform information gathering. The chapter is structured as follows: Section 5.1 performs a brief introduction. In Section 5.2, we introduce the relevant background and related works. Then, in Section 5.3, we detail the proposed enhanced response and information gathering mechanism. Section 5.4 critically evaluates the results from the PoC. Finally, Section 5.5 presents the conclusions and significant findings.

5.1 Introduction

The tighter integration of business verticals into shared network infrastructures, as in the 5G Vertical Slicing use-cases, raises significant security, trustworthiness, and reliability issues. Such issues are particularly relevant in the context of Industry 4.0 and the IoT, where a large number of embedded systems that cannot run advanced anti-malware routines are exposed to that shared infrastructure.

One way to thwart the cyberattacks against Industry 4.0 and IoT infrastructure is deploying honeynets [121]. A honeynet is an isolated sandbox network with decoy functions (honeypots) mimicking the protected functions. The honeynet approach allows more advanced intelligence gathering about the attacker's objectives, motivations, and exploitation techniques while still thwarting the attack's adverse effects over the actual network slice. The decoy's effectiveness varies according to the application being mimicked and the level of interaction with the attacker. However, before reaching the decoy and deliberating about its effectiveness, two things must happen first: (1) the attack must be detected and (2) the network must steer the active attack seamlessly.

Figure 5.1 and Figure 5.2 show how to steer an undergoing active attack into a sandbox network slice with a decoy network function. Figure 5.1 illustrates the system's initial state where the attacker interacts with the actual network slice. In order to detect the attack, there will be some form of IDS/IPS that will inspect the interactions with the real NF and detect known attack patterns, malware fingerprints, or contextually anomalous activities. Upon detecting the attack, an alarm will be sent to the network slice's management layers, identifying the offending flow and signaling the need to steer that flow to the sandbox slice. The slice management layer should then use network control such as

SDN to enforce the steering policy and effectively redirect the attack into the decoy NF (as shown in Figure 5.2).

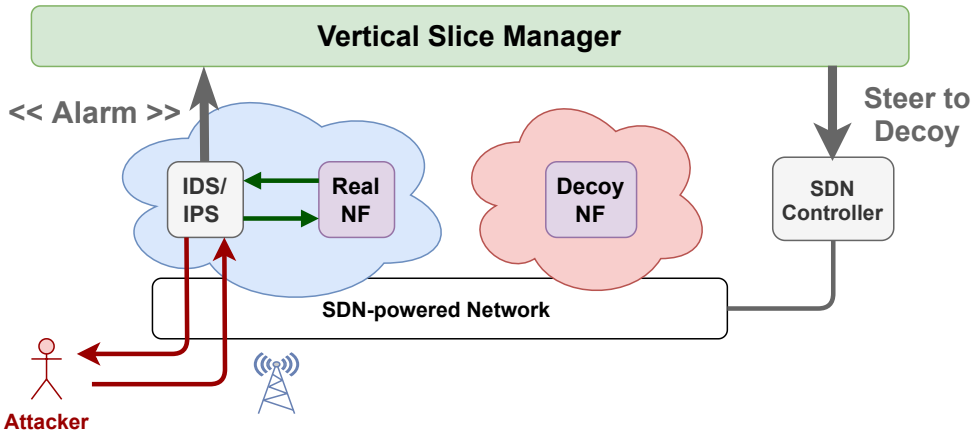


Figure 5.1: The initial system state, the attacker interacts with the Real NF

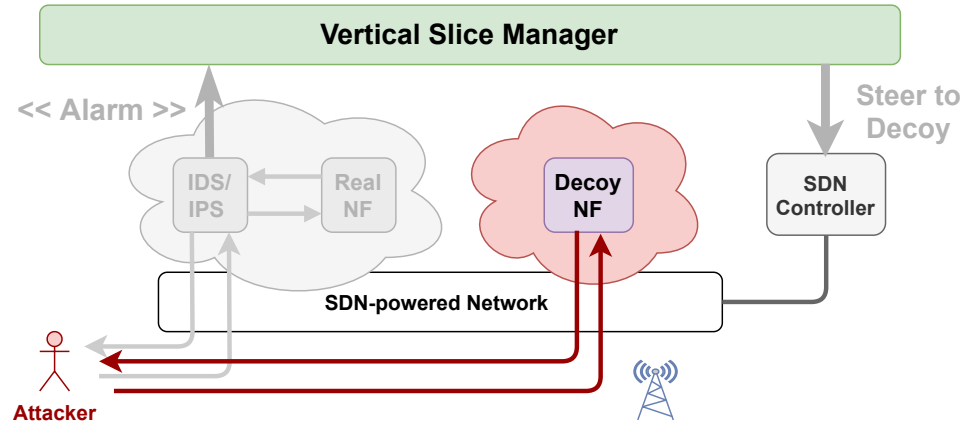


Figure 5.2: The desired system state, the attacker is seamlessly steered to the Decoy NF

The TOTP-MTD mechanism discussed in the previous chapter (Chapter 4) is a drop-in IDS/IPS replacement that can detect adversarial action more effectively for this scenario. Because the TOTP-MTD mechanism does not rely on previous attack patterns or heuristics, it is as effective against zero-days as other previously known attack types. The IDS/IPS block is shown here instead of the TOTP-MTD contribution to avoid losing generality in our solution. However, while the MTD mechanism effectively addresses the detection part of the pre-conditions, it does not solve the required endpoint steering to the decoy. TOTP-MTD moves the same endpoints across the exploration space but does not move the connection across different endpoints.

Slice-based Network Control already enables the isolation required for those sandboxing networks, but the existing traffic steering capabilities and mobility approaches only address a part of the attack steering problem. The main issue is that TCP relies on the parameters negotiated during the three-way handshake to establish the connection, such as the successively incremented sequence number to acknowledge each packet (as shown in Figure 5.3). Merely outputting the following packets of the flow to the new endpoint will not suffice to achieve any interaction with the decoy (as crucial TCP session information, such as the correct sequence number, is missing to communicate).

The SDN controller can bootstrap a smooth handover of an active TCP session across endpoints, being the core of a new mechanism. This purpose-built proxy function will resume the active attack

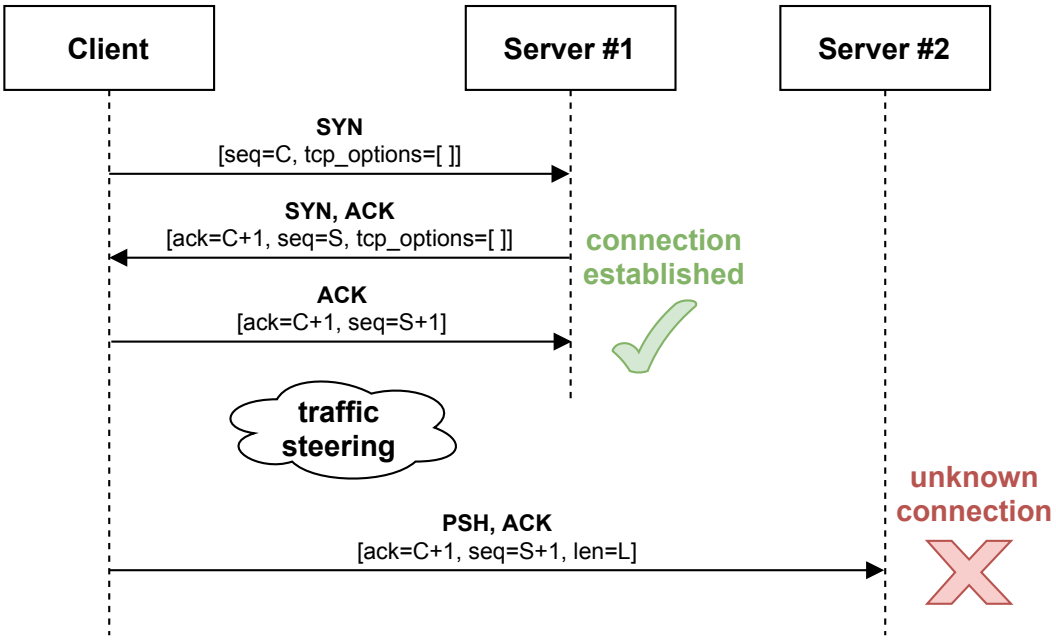


Figure 5.3: Illustrating the issue with steering an active TCP session to a different endpoint

session using the `TCP_REPAIR` features of the Linux Kernel. Because it is effectively recreating the socket as if the connection was initially established with that new endpoint, all of the TCP session state machine and control sequence inner-workings are done seamlessly by the kernel's built-in routines and the higher-level abstractions that use them.

5.2 Background and Related works

This section will introduce the security enhancement use-case of this thesis's endpoint handover mechanism, the enablement of honeynets as a foundational building block against the attackers' active offensive activities. It will briefly complement the background information in Chapter 2 with the specific shortcomings of the current network slicing technologies to support this use case. Then, it complements the related works in Chapter 3, present how other works have resolved these challenges and critically relate our mechanism's novel aspects to existing approaches.

Honeynets are isolated honeypot (decoy) networks used to detect new attacks, as an intelligence-gathering tool of the current attackers' techniques and, as a general way to prevent damage to the real networks [121]. Their level of interaction classifies honeypots: from Low-Interaction Honeypots (LIH) that only perform the most basic liveness routines of an endpoint (*e.g.,* pretend an IP is in use, when in fact it is not), passing by Medium-Interaction Honeypots (MIH) that simulate the responses to more superficial interactions with some services of the network stack (*e.g.,* a DHCP lease), going all the way to Highly-Interaction Honeypots (HIH) that feature convincing endpoint emulation (*e.g.,* not just more complex responses that may rely on service state, but may also interact with its Operative System and other internal processes).

A significant requirement for the proper operation of a honeynet is good traffic steering capabilities. The traditional steering techniques in Network Slicing include simple flow rules that redirect traffic at the switch level (*i.e.,* change output port and addresses), the Telco-grade traffic engineering of MPLS networks, or the newer steering techniques from SFC such as the Network Service Header (NSH) [122]. However, none of these solutions allows us to successfully steer an active TCP connection to another

destination endpoint, hindering honeynets' dynamic and on-demand use after detecting an attack in the production network.

To solve this issue, *Binder et al.* [123] proposed an SDN based method of TCP handover that could operate in two modes: *(1)* using the SDN Controller directly as the frontend to interact with the active TCP session and mangle the packets within the controller, or *(2)* deploying an additional software switch within the destination endpoint that was modified to perform the packet mangling functions. The modifications to the software switch and its operational needs resulted in a version of the OpenFlow 1.3 protocol with additional non-standard `SET_FIELD` actions to control the TCP packet mangling offloading. *Fan et al.* [124] proposed a similar mechanism to *Binder et al.* [123] aimed directly at honeynet. They have used OpenFlow's standard experimenter messages to control those custom switch actions, therefore avoiding the dependency on a non-standard version of the OpenFlow protocol as in *Binder et al.* [123]. Building upon their work, *Fan et al.* later introduced the HoneyDOC architecture [125], featuring their seamless TCP handover mechanism alongside the honeypot functions' management and orchestration. In the work proposed in this thesis the HoneyDOC approach the TCP handover mechanism using just the SDN controller as the frontend. It was opted to refer to HoneyDOC instead of the preceding *Binder et al.* work simply because there is a much more significant overlap with our mechanism's security enhancement use case. Despite the overlap, our work critically diverges from *Binder et al.* [123] and *Fan et al.* [124][125] by continuously replacing the SDN Controller's TCP packet mangling with a proxy function that can be instantiated and chained on-demand. Furthermore, the proposed proxy function exposes more flexible NFV interfaces instead of relying on direct SDN control through the OpenFlow protocol, like the optional modified switch in these works.

Other SDN-based approaches already exist, such as the HoneyProxy [126] and HoneyMix [127]. Unlike the proposed mechanism, the HoneyProxy approach requires that all TCP connections be established through the proxy. That design pattern is less flexible, likely more resource-demanding, and a potential central point of failure. In turn, HoneyMix assumes that a live TCP session steering mechanism already exists.

The Honeybrid [128] framework solves this issue using a gateway that employs Linux netfilter (aka "iptables") to select flows and then `NFQUEUE` to mangle packets in a user-space program. The approach is effective but does not offer smooth integration with Network Slicing management or SDN control.

The mechanism proposed in this thesis allows for the best of both worlds. It enables the opportunistic steering of a detected attack in the real network while still avoiding extensive packet mangling in user-space by using the Linux Kernel's `TCP_REPAIR` socket options, thus preserving the same kernel-space routines of the networking subsystems as in a regular socket (*i.e.,* an expected performance advantage against the previous solutions).

5.3 TCP_REPAIR Solution and Architecture

This section will present the solution that comprises the endpoint handover mechanism of this thesis. The mechanism's main novelty is a purpose-built proxy leveraging `TCP_REPAIR` (aptly named "`TCP_REPAIR` Proxy") to resume an active TCP session with a new endpoint seamlessly. However, the proxy only performs the required TCP-level changes to continue the session, not the prerequired connection steering. Therefore, this connection steering requirement will be approached, mainly how to tackle the active connection's endpoint handover. To do so, an overview of the architecture that integrates this proxy solution into the network slicing environment will be presented, and then discussion will go deeper into how the handover process works within that architecture.

5.3.1 The TCP_REPAIR Proxy

To enable the steering of an attacker into a Decoy NF, we must first solve the challenge of modifying the already established TCP session with the Real NF in a way that makes it usable with the Decoy NF (even if the communication is already mid-way). The approach is to insert a custom proxy server between the attacker and the Decoy NF, which will take over the attackers' existing session with the Real NF, and then forward the payloads back-and-forth to the Decoy NF through a newly established regular socket. This proxy was called the `TCP_REPAIR` Proxy because it leverages the `TCP_REPAIR` features of the Linux Kernel[1].

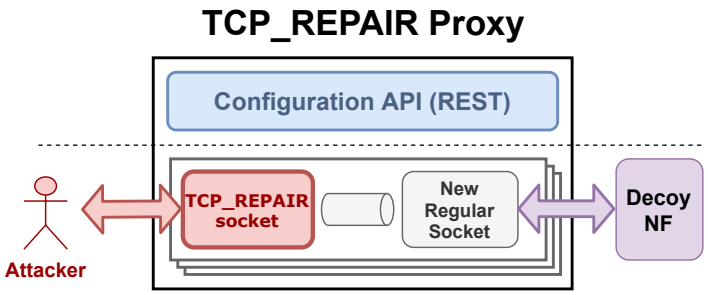


Figure 5.4: The TCP_REPAIR Proxy approach

The `TCP_REPAIR` Proxy (shown in Figure 5.4) behaves as a regular TCP proxy in the forwarding plane. The significant distinction is its capabilities to instantiate TCP sockets at any stage of their state machine. That is, the three-way handshake is no longer needed to establish a connection successfully but instead may create (or resume) TCP sessions at any desired state with any pair of endpoints. Figure 5.4 denotes the socket with these capabilities as the `TCP_REPAIR` *socket*. Instantiating this socket and recreating an active session requires a configuration API through which the proxy receives the necessary TCP state to resume the connection seamlessly. The state information must include the IP addresses, TCP ports, and each endpoint's latest TCP sequence numbers. The proxy can also restore other parameters of the already active session (TCP options), even those negotiated during the three-way handshake, such as the Maximum Segment Size (MSS) or the window scale factor, provided that those values are made available through the configuration interface. The MSS is a crucial parameter to avoid breakages, such as when the attacker requires a lower segment size than our default, thus causing packet drops or fragmentation. In turn, the window scale factor is significant for the line-rate performance and crucial to maintain stealthiness (*i.e.,* not reveal that the session was redirected to another endpoint).

Simultaneously with the `TCP_REPAIR` *socket*'s recreation, a new regular socket is established from the proxy to the Decoy NF. After that, all communications within that active session will be piped through the two sockets, like a regular proxy. The design allows the instantiation of multiple socket pairs so that a single proxy instance can serve multiple connections to the same decoy.

The approach's advantage is that, once the `TCP_REPAIR` *socket* is recreated, the performance should be nearly the same as if the connection was initially established with this proxy through a regular socket. That is, it should benefit from the lower-level packet handling of operating systems' underlying networking subsystem, rather than overriding these subsystems with user-space packet mangling code such as `NFQUEUE` or having the SDN Controller as a frontend (as seen in the approaches of Section 5.2). Because the `TCP_REPAIR` is a socket option readily available in higher-level programming languages

[1] https://lwn.net/Articles/493983/

(such as Python), this approach yields the best of both worlds: it is easier to program more advanced features while still delivering the same high performance of the underlying networking subsystem.

The trade-off is that, like any other proxy, the bottleneck is likely the applications' piping performance between the two sockets. We will make sure to assess this parameter against other approaches in the evaluation section. Additionally, because the session steering is performed outside of the proxy, the solution also requires the proper synchronization of the socket configuration parameters with the latest flow packets to resume the session successfully. This synchronization was achieved using the handover approach described in the following subsection.

5.3.2 The Handover Architecture

The previous subsection presented an approach to the challenge of modifying an already established TCP session with the Real NF in a way that makes it usable with the Decoy NF (even if the communication is already mid-way). Now it must be considered the coordination of the steering in the SDN network with the `TCP_REPAIR` Proxy configuration to successfully steer the active connection. Figure 5.5 illustrates the challenge at hand: there are two separate components whose configuration depends upon a live TCP connection, and the values set while configuring one (*e.g.,* the sequence numbers sent to the proxy) must still be valid after the other configuration request completes (*e.g.,* the steering by the SDN controller).

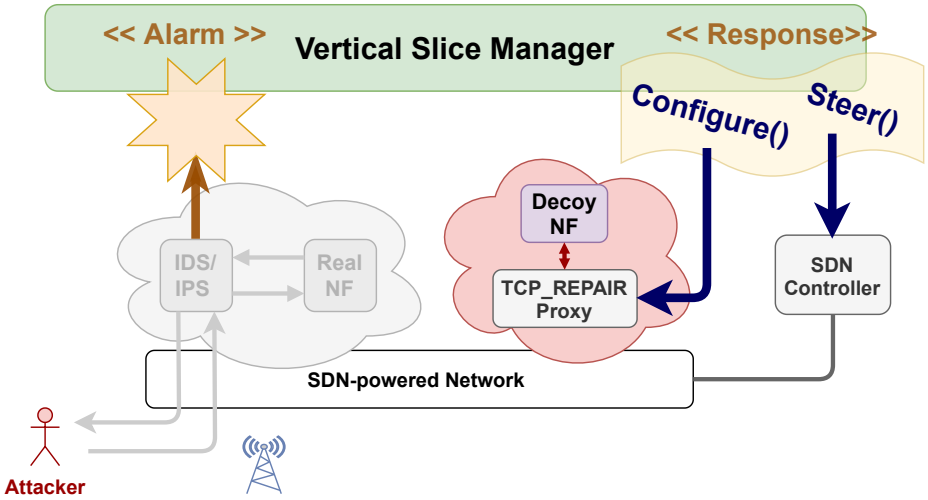


Figure 5.5: Overview of our TCP_REPAIR Proxy approach

Going further into the details, there is an alarm event that flags a given TCP connection for steering, but doing so without breaking the connection, requires that we first configure the `TCP_REPAIR` Proxy. Only then can the packets be steered towards the Decoy NF. Without a loss of generality, an alarm event originating from an IDS/IPS system is illustrated. Because the following TCP packet parameters must match the recreated `TCP_REPAIR` socket configuration, especially the latest sequence number of each endpoint, a handover process is required instead of just direct steering after enabling the proxy function. The handover process must temporarily take over the offending flow so that we have no packet loss, double delivery (*i.e.,* both the Real and Decoy NFs receive the same packet), or loss of synchronization (*i.e.,* sequence numbers match the configuration in the proxy). To do so, the SDN Controller will be the central synchronization point of the handover process.

Some of the crucial session parameters are negotiated between the endpoints during the three-way handshake that precedes establishing the connection. Therefore, gathering of that information must

start, well before any malicious payload is detected and an alarm event is generated — at that moment, the connection would already be established; thus, the handshake parameters would be irrecoverable without the collaboration of the endpoints. This issue was solved by having the controller intercept all three-way handshakes and record the negotiated parameters of the active TCP connections. This interception is easy to achieve with OpenFlow 1.5 as the protocol supports sending to the switch(es) flow rules matching the relevant TCP flags (SYN, RST, and FIN). Older versions of the OpenFlow protocol can still perform this interception by installing a rule that matches all TCP packets and sends them to the controller. However, this comes at a significant efficiency cost. Upon completing the handshake, the flow may resume its forwarding-plane-only path.

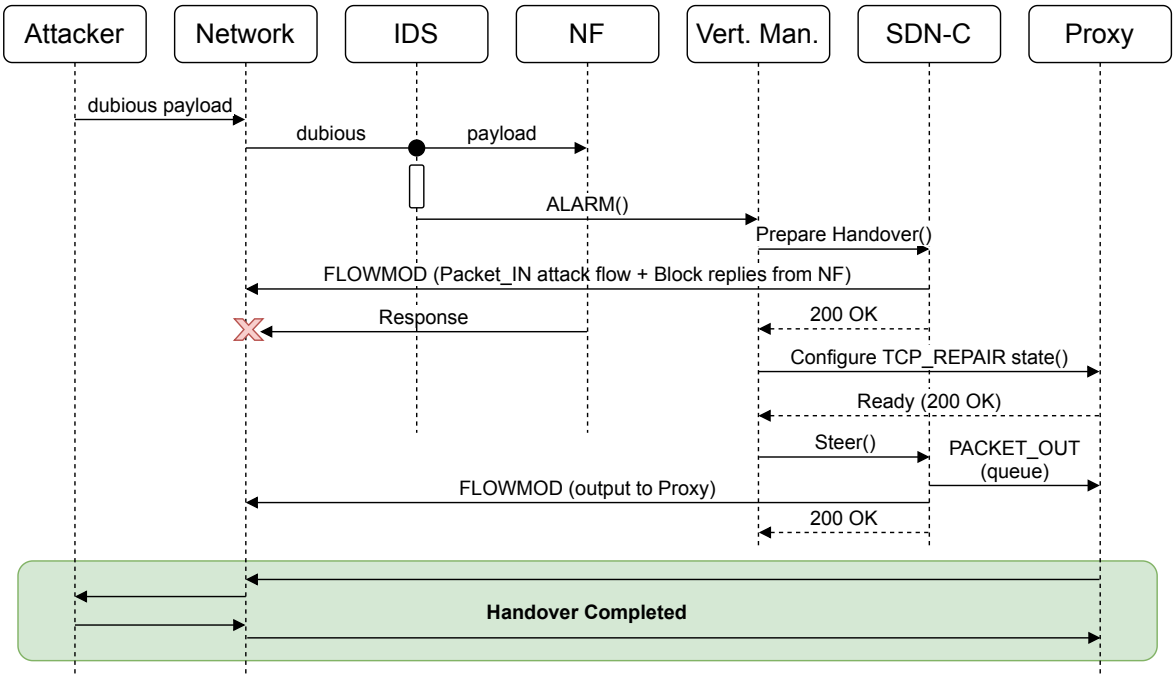


Figure 5.6: Handover Sequence Diagram

Thanks to this TCP session tracking in the SDN controller the handover process was modeled accordingly to the sequence diagram shown in Figure 5.6. The Vertical Slice Manager receives an alarm event once the IPS detects an attack (or abnormal activity). Upon receiving the alarm, the Slice Manager notifies the SDN controller to prepare for the handover. That means the SDN Controller will intercept the offending flow by issuing flow rules that `PACKET_IN` all new packets originating from the attacker and blocking all new replies from the Real NF (preventing further data exfiltration). The SDN controller starts queuing the new packets of that offending flow, gathering the next packet's sequence number, and returning those parameters to the Slice Manager in the "200 OK" response. This handover stage allows freezing the packets in the active connection until the `TCP_REPAIR` proxy is configured, thus achieving a properly synchronized session configuration (especially the sequence numbers). Once the Slice Manager receives the "200 OK" response that confirms the proxy is configured, it sends a steer request to the SDN controller so that the enqueued packets of the active TCP session are now released and sent to the Decoy NF via the proxy. The release of enqueued packets is done with `PACKET_OUTs` until the queue is empty, at which point the steering request is finalized by issuing the required `FLOWMODs` that will replace the MAC and IP address of the Real NF with those of the `TCP_REPAIR` proxy in the offending flow. From hereafter, the live-session resumes, and the handover is completed.

5.4 Evaluation

This section will evaluate the `TCP_REPAIR` Proxy approach within a PoC that implements the architecture described in Section 5.3.2. The evaluation was performed in two ways. First, it was experimentally validated if the solution fulfills its active TCP session steering objectives and qualitatively evaluated the findings. Then, the solution was benchmarked in the crucial parameters for network slicing and the results were evaluated quantitatively.

5.4.1 Qualitative evaluation

We will begin by validating that our solution works as expected. We will then evaluate qualitatively how well the architecture and interfaces integrate with our network slicing PoC environment.

To validate if the `TCP_REPAIR` Proxy approach worked, we have used the *GNU Netcat*[2] tool (aka *netcat-traditional* in Debian-based distributions) to simulate the client and the two servers (real and decoy). This netcat tool is widely used in the testing of the shellcode of binary exploits and, much like the server shellcode of those exploits, it lacks a more advanced connection recovery mechanism — by default, once there is an issue in the already established TCP session (*e.g.,* if the connection needs to be reset), the server die and will not allow for recovery.

We have tested the *null hypothesis* using the *netcat* tool to establish a connection between the attacker and the real server. Then, we used the SDN controller to do a simple steer of the connection to the decoy. As expected, we verified that *(1)* the attacker could no longer contact the real server, and *(2)* the decoy could not interact with that session despite receiving the packets with proper addresses (IP and MAC matched the decoy endpoint). The client would eventually error out with a connection issue, potentially alerting the attacker that something abnormal had just happened. We proceeded to test the case where the attacker attempts to establish a new connection to recover from that connection error. Because the steering rules were already in place, everything now worked as originally intended — the attacker was now interacting with the decoy server. This outcome validates that the traffic steering was done correctly, new TCP connections would work fine, but the challenge of steering an active session did not (as expected by the *null hypothesis*). For completeness sake, we have also validated the expected shellcode-alike behavior of the *netcat* servers. The actual server no longer accepted new connections, as it already had received the connection we were steering. Similarly, the decoy server would not accept new connections after the attacker's reconnection attempt. Therefore, we have easily reproduced in our testing scenario the challenge that motivated our `TCP_REPAIR` Proxy solution.

After resetting the experiment, we have then proceeded to the validation of our `TCP_REPAIR` Proxy. As the *null hypothesis*, we have started using the *netcat* tool to establish a connection between the attacker and the actual server. Unlike the *null hypothesis* that proceeded to steer the traffic directly, we will now use the SDN controller to perform the handover, as described in Section 5.3.2 (and shown in Figure 5.6). We have confirmed that the TCP session data tracked by the SDN controller matched the parameters reported back to the Slice Manager and that the `TCP_REPAIR` Proxy was then configured with the correct values. As the `TCP_REPAIR` Proxy was being configured and the new sockets were instantiated, we could observe a new connection in the decoy (as expected). Upon completing the handover process, there was no apparent error in the client, nor any indication that the connection had been steered to a different endpoint (from the real server to the decoy). Finally, we have sent data back-and-forth from the client to the server mimicking an attack's interactions. The TCP session

[2] http://netcat.sourceforge.net/

remained open and could send and receive data to the decoy, validating that the `TCP_REPAIR` Proxy successfully solved the challenge.

The new configuration interface exposed by the `TCP_REPAIR` Proxy integrates neatly into the slicing environment as a regular NFV-MANO primitive, much like other NFV-enabled network slice functions. Similarly, the new proxy function (our main contribution) is easily instantiated, much like other NFV functions. Moving to the steering capabilities, these are also neatly exposed through the controller's northbound interfaces (in our case Ryu[3]), much like similar steering functions already endorsed by some other SDN controllers (such as ONOS[4]).

In turn, although the functionalities required to capture the three-way handshake information of all active TCP sessions in the protected network are straightforward to implement and use, our mechanism requires OpenFlow 1.5 for better efficiency. Unfortunately, some of the commercially available enterprise switches do not support this version. Nevertheless, the widely popular Open vSwitch[5] supported this OpenFlow version and was the switch of choice for our tests. Lastly, the handover functionality requires add-ons both to the SDN controller and the Vertical Slice Manager. In the SDN controller case, the handover add-on is not more complex than the steering or active connection tracking add-ons. As for the Vertical Slice Manager, the logic required is a straightforward event-driven management routine, likely far less complex than the routines that already assure the KPIs' continuous delivery.

5.4.2 Quantitative evaluation

To evaluate our `TCP_REPAIR` proxy approach, we have implemented the PoC as described in Section 5.3.2 and compared the `TCP_REPAIR` Proxy approach's performance against other approaches (introduced in Section 5.2) – notably the honeybrid and the HoneyDOC. In order to eliminate the contribution of implementation-specific characteristics, such as older frameworks or language-specific optimizations, we have reimplemented the relevant parts of those approaches using the same Python interpreter as our proxy. By doing so, we will be able to fulfill the main objective of this evaluation: an apples-to-apples comparison of our `TCP_REPAIR` approach vs. `NFQUEUE` (as in honeybrid) vs. having the SDN Controller as a packet mangling frontend (as in HoneyDOC).

The comparison focuses mainly on latency and throughput because these were the metrics that yielded different results. Other parameters, such as CPU load or memory usage, were also considered in the experiments. However, all reimplementations herein evaluated will take 100% of a CPU core when under load and consume similar amounts of memory for the same workload; hence these other metrics were not highlighted in this quantitative evaluation. In effect, the key differentiation was the latency and throughput achievable by each approach when under load. In order to understand the significance of these numbers, all tests were repeated in the same environment against the regular communication using plain TCP sockets (without steering or any other change) — we have called this the baseline for our tests.

We will start with the latency evaluation. We have built a custom python client and server that effectively measure the Round-Trip Time (RTT). Because the PoC is in a controlled environment with symmetrical paths both for send and receive, we believe it would be reasonable to infer that latency is half of the RTT. The custom client uses a TCP socket to send a timestamp with precision up-to-the microsecond, which gets echoed by the server. The client then calculates the RTT by subtracting

[3]https://ryu-sdn.org/
[4]https://www.opennetworking.org/onos/
[5]https://www.openvswitch.org/

the system clock truncated to the microsecond with the received echo server's timestamp. All test cases were repeated at least 10 000 times. The results displayed in Figure 5.7 only consider the 95% confidence interval values, and the bars show the calculated average with a vertical line depicting the distance to the minimum-maximum. The red dotted line is an extension of the baseline experiment to facilitate the reading of the graph.

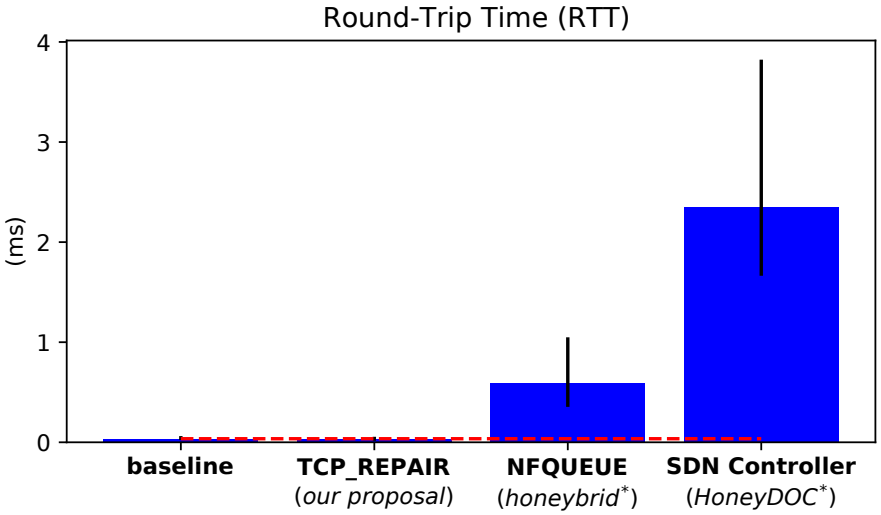


Figure 5.7: The measured RTT of each of the TCP sequence mangling approaches

The results show that the `TCP_REPAIR` approach performed on-par with the baseline at nearly 0.03 milliseconds. We consider this an ideal result for network slicing. The `NFQUEUE` solution (honeybrid) performed much worse, having nearly 0.6 milliseconds of RTT. The SDN Controller as the frontend (HoneyDOC) performed the worst, averaging at nearly 2.35 milliseconds. Network slicing may demand support for low-latency scenarios (*e.g.,* $< 1ms$ of 5G), therefore the competing approaches' results are impactful and potentially voiding usage under such network slicing scenarios.

Moving on to the throughput evaluation, we have created a custom python client that recorded the time that took sending a large file to the server over the TCP socket. The server was a python program that just received the data (without echoing it). The recorded throughput is a simple division of the file size over the transfer time. All test cases were repeated at least 120 times. The results displayed in Figure 5.8 only consider the 95% confidence interval values, and the bars show the calculated average with a vertical line that depicts the distance to the minimum-maximum. The red dotted line is an extension of the baseline experiment to facilitate the reading of the graph.

The throughput results show that the `TCP_REPAIR` approach performed the closest to the baseline, at nearly 588 Mbit/s (vs. 615 Mbit/s of the baseline – an average loss of less than 5%). The alternative approaches fared much worse than the `TCP_REPAIR` approach. The `NFQUEUE` solution (honeybrid) achieved a net rate of nearly 42 Mbit/s, an average loss of well over 90%. Like in the RTT experiment, the SDN Controller as the frontend (HoneyDOC) performed the worst, reaching nearly 15 Mbit/s of throughput. The experimental results show that the `TCP_REPAIR` approach is superior when higher-throughput is a deciding factor (*e.g.,* enhanced Mobile Broadband (eMBB) in 5G). However, both the `NFQUEUE` (honeybrid) and SDN Controller as the frontend (HoneyDOC) may still be serviceable in other more common sandboxing network slice scenarios, where the attacker's interactivity with the decoy may not require that much throughput (*e.g.,* a remote shell).

The quantitative results show that our proposed `TCP_REPAIR` approach had the best performance

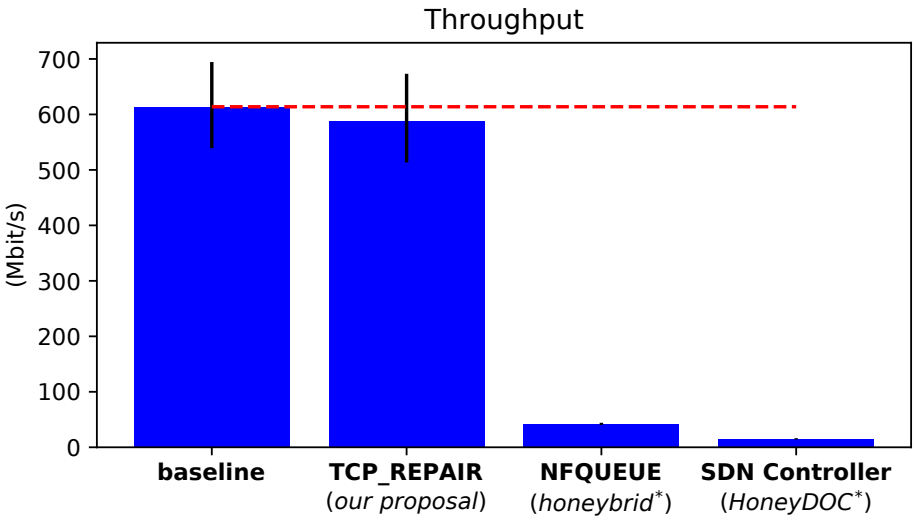


Figure 5.8: The measured throughput of each of the TCP sequence mangling approaches

for the various network slicing use-cases. Because the `TCP_REPAIR` approach performed the closest to the baseline, it is also likely the better choice for stealthiness (*i.e.,* avoid detection by the attacker).

5.5 Summary

We have effectively used the `TCP_REPAIR` socket options built into the Linux Kernel to devise a proxy solution that can receive an active TCP connection (*e.g.,* an attack) and seamlessly continue that session with a new endpoint (*e.g.,* the decoy). Our solution's performance was near the baseline of a regular socket, with the latency nearly the same. At the same time, the average throughput was reduced by less than 5% of the regular socket. That was a great result, especially when compared to the alternative approaches of `NFQUEUE` (honeybrid) and the SDN Controller as the packet mangling frontend (one of the operating modes of HoneyDOC), that affected the latency and the throughput very significantly (different orders of magnitude in throughput).

The `TCP_REPAIR` Proxy handles the live TCP session piping to the new endpoint, not the handover required to steer the traffic. Therefore, we have devised a handover sequence that leverages the SDN controller and integrates it nicely into the Slice Manager to solve this issue. We have experimentally validated that our handover approach worked correctly and gathered all performance data from the `TCP_REPAIR` Proxy already using this handover solution. We leverage standard features of the OpenFlow 1.5 protocol to match specific TCP flags to gather (via `PACKET_IN`) the critical three-way handshake data (*e.g.,* negotiated Maximum Segment Size, window scale factor, and sequence numbers) and keep track of the active TCP connections. Using a previous OpenFlow version will come at an efficiency loss, as gathering this data would require sending all TCP packets to the controller.

Lastly, we have qualitatively assessed our solution for use within network slicing systems and concluded that the integration footprint would not be of higher complexity than the already existing modules.

CHAPTER 6

Optimization and KPIs

This chapter researches a straightforward way for business verticals to optimize the computational resources usage and potential security benefits within an MTD framework, in particular one that uses the main contribution described in Chapter 4. The chapter is structured as follows: Section 6.1 performs a brief introduction. Section 6.2 introduces the relevant background and related works. Then, Section 6.3 details the proposed security as a KPI approach. Section 6.4 critically evaluates the results from the PoC. Finally, Section 6.5 presents the conclusions and significant findings.

6.1 Introduction

Network slicing is an infrastructure virtualization technology that allows creating isolated virtual networks, called slices, with specific requirements and functions. NFV is a carrier-grade cloud computing framework that focuses on the management requirements of the telecom world. Both Network Slicing and NFV are enablers of the 5G networks.

The requirements of a network slice are set using higher-level descriptions of the desired outcome, called KPIs, which are meant to be measurable and continuously assessable. Bodies such as the GSM Alliance have proposed using as KPIs the latency, throughput, power consumption, security, and others. However, while latency, throughput, and power consumption are mensurable universally, security is more complex to measure [129]. For starters, security is multidimensional, so measuring security would undoubtedly require a set of metrics for each attribute in consideration [129]. Each metric would have to fit its intended purpose, audience, and goals, which are tightly coupled with the function being served and its specific threat model [129]. While such a set of metrics would be the more accurate description of the systems' security, untrained business verticals could quickly find the multitude of metrics a barrier to the adoption, or worse, become counterproductive if wrongly specified.

Thus, the goal is to establish a straightforward network security metric that allows the network slice provider to deliver some level of security regardless of the actual service being protected, or the proficiency of the business vertical specifying that requirement. To do so, newer proactive defense approaches, such as MTD [36] are explored. The core principle of MTD is to explore the available network configuration space dynamically (*i.e.,* create movement) to frustrate the attackers' endeavors. Provided the attacker does not know the secret that creates the movement, the MTD mechanism may either break the connectivity of the attacker to the service, forge data (or interactions) so that the attacker acts upon wrong intelligence or a combination of both. This approach is effective because

a successful attack and exploitation of a target require that the attacker must first perform the reconnaissance of the virtual surroundings, identifying the objectives, along with the vulnerabilities of each service that may lead to its goal [24]. Therefore, this thesis proposes to derive a comprehensive metric of Dynamic Security (*DynSec*) by disrupting the attack process at the very start, using a measurable defense mechanism and straightforward discrete math.

6.2 Background and Related works

It is vital to relate the proposed metric with the existing security quantification methods. It is also relevant to relate the identified shortcomings of the security quantification methods with the needs presented by network slicing. Lastly, it is important to introduce some of the MTD mechanisms that create movement within the exploration space, upon which our proposed metric is built.

Security is multidimensional, so measuring security would require a set of metrics for each attribute in consideration that is dependent on its intended purpose, audience, and goals [129]. *Xu S.* [130] thoroughly surveyed security quantification in the Cybersecurity Dynamics proposal, outlining different metrics per representation of security being measured. The general model identifies four key measurements: *(1)* the attack-defense structure, which is the connectivity between attackers and defenders, *(2)* the susceptibility of the systems, *i.e.,* the exposure and vulnerabilities, *(3)* the defense capabilities, which is the effectiveness in detecting or stopping an attack, and *(4)* the attack capabilities, a representation of the proficiency of the attacker in evading detecting and exploiting vulnerabilities. Our proposal focuses mostly on the attack-defense structure and the defense capabilities of the MTD approach to delivering a single simplified metric of security that does not depend on a thorough knowledge of the network services being protected, as required to assess (and repair) the susceptibility of the systems. This simplification lowers the barrier of access to comprehensive and quantifiable basic security and is especially suited for the scenarios where no security considerations would be made otherwise. However, we must stress that this proposal does not claim (or even attempt) to replace the existing multidimensional security quantifications.

While such a basic network security metric might be useful in other networking scenarios, network slicing emerges as the primary use-case for a good reason. New slicing-enabled networks such as 5G are bringing the business verticals into roles that were traditionally enclosed inside the telecom world (*e.g.,* deploying functions within the telco infrastructure, having access to some private interfaces), increasing the telco exposure to attacks while creating an immediate skills gap. Network slicing already specifies various KPIs to characterize the service within a slice, among which is a security KPI. Because of the multidimensionality of security mentioned above, that security KPI would be far from universal, therefore harder for a provider to continuously assess its correct delivery. Quantifying security in that manner requires in-depth knowledge of the services inside the slice and purpose-built monitoring. These difficulties could become a barrier for adoption, or worse, an incentive to disregard security. Because of this, it is proposed using more universal defense approaches such as MTD, and then measure their effectiveness.

There are a plethora of MTD mechanisms that deliver some form of network defense through the mutation of various service parameters, using different approaches and technologies depending on the scope of application. *Kewley et al.* [57] proposed DYNAT, a transformation function of the destination IP address and port that used an encryption algorithm (RC5), along with a secret seed and a time-based secret key. Because of this transformation, only those who held the shared secrets could reliably contact the services behind DYNAT. Those who did not would be misdirected to a

random IP address and port pair, that directly resulted from a transformation with the wrong secrets. This misdirection would make network scans take longer while still being harder to return actionable information, also aiding in the detection of attacks such as DoS (higher traffic dispersion than the limited valid destinations), and putting a time limit on a session hijack. *Jafarian et al.* [59] introduced the OF-RHM technique. Unlike DYNAT, there is no need to pre-share secrets in order to find the end-point, as new DNS responses are created according to the current mutation. A random virtual IP address will be assigned from a pool of available addresses, being this random function either uniform or weighed (according to the chosen mutation type). The service end-point will have a real IP address that does not move, being the random virtual IP address a moving translation to the matching real IP of the service. Other proposals, such as NASR by *Antonatos et al.* [60], allow mutating the IP address of the service by changing the responses in the DHCP protocol. Unlike the previous methods, which are direct translations of serviceable end-points, *Sun et al.* [64] proposed DESIR, which introduced deception on top of the previously discussed mutations as a means to frustrate attackers. Later works started providing multiple mutations, creating more uncertainty for the attacker. The MUTE [71] architecture, on top of fingerprint mutation, also allows mutating the IP addresses. MUTE's early work in network reconfiguration for MTD would become a reference for the later works which leveraged emerging network configuration technologies. Proposals such as SDN-MTD [69] and CHAOS [75] used SDN's abilities to mutate IP addresses, ports, and fingerprints. The proposal in this chapter builds atop similarly capable mechanisms as the ones referenced to construct a straightforward metric of dynamic network security, which is easier to quantify and use as a basic security KPI of a network slice. Crucially, the KPI does not establish a Service Level Agreement (SLA), but rather just the benchmark. The SLA states the specific amounts of that benchmark, along with when and how these amounts must be delivered.

In sum, after reviewing the existing security quantification methods, there is a lack of a straightforward security KPI due to the multidimensional nature of security. However, the large number of relevant MTD mechanisms already available presents with the opportunity to design a new kind of basic security metric, which will be presented in the following section.

6.3 Creating Mensurable Network Security

The foundational idea of this new metric is that an attacker cannot compromise or even gather information if it cannot establish a connection with the target. Taking the analogy of a shooting-range target, we will categorize as a *hit* when the network service is reached, and as a *miss* when it is not. We consider the service remains secure if the attacker *misses* it. Conversely, because of the unidimensionality of this model, the service is insecure if the attacker can *hit* it. While there may still be additional security features built into the function(s), those are not a part of this basic network security as a service feature.

Let it be assumed that the attacker starts the reconnaissance process without prior knowledge of the service configurations running in that network, and that each probing attempt checks a previously unprobed network endpoint configuration (*e.g., protocol, ip,* and *port* tuples). Because network services usually remain serving with the same configuration their entire lifecycle, the probability of getting a *hit* in the n-th probing attempt is:

$$P_{hit}(n) := \begin{cases} \frac{\binom{total\_configs}{n} - \binom{total\_configs - active\_configs}{n}}{\binom{total\_configs}{n}} & , \; n \leq (total\_configs - active\_configs) \qquad (n \in \mathbb{N}_0) \\ 1 & , \; n > (total\_configs - active\_configs) \end{cases}$$

$total\_configs$ denotes the maximum available network configurations and sets the MTD's exploration space. $active\_configs$ denotes the currently used network configurations and sets the attack surface. $active\_configs$ can never exceed the $total\_configs$, but it can match the same value or be zero. Assuming we have random network configurations used within the known exploration space, $P_{hit}(n)$ denotes the probability of hitting the attack surface.

Without a loss of generality, if we exemplify with a single TCP/UDP listening port mutation mechanism, we have a well-defined limit of $total\_configs = total\_ports = 2^{16} - 1 = 65535$. The relatively small number of total combinations, along with the fact that an attacker could parallelize the scanning process, translates to that *hit* being achievable in a reasonably short timeframe. The smaller configuration space is a conscious decision to better illustrate how this metric would work. Once the service is found, the attacker starts collecting more information, using different probing techniques to find potential vulnerabilities. Time is on the attackers' side to find and exploit a vulnerability, while network function maintainers have time playing against their security designs; this effect of time is known in the MTD field as the asymmetric relationship between attackers and defenders.

The entry points for the reconnaissance process vary depending on the existing foothold in the network, as shown in Figure 6.1 and Figure 6.2. When the attacker has no foothold, it will have to use just the public interfaces for the reconnaissance (Figure 6.1). However, powerful state-backed attackers may already have some form of illegitimate connectivity to a management network in the NFV powered network slicing system (Figure 6.2), for instance, through implants in the network equipment while it was being shipped.

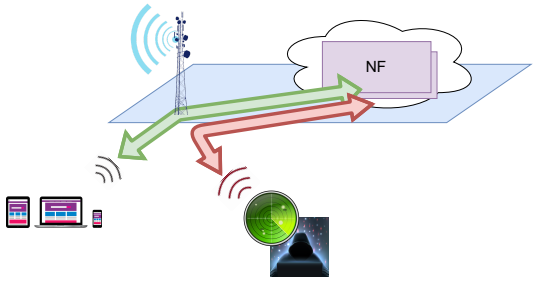


Figure 6.1: Reconnaissance in an NFV powered Network Slice using public interfaces

Firewalls and more advanced IDS/IPS have been effective in detecting and stopping reconnaissance [131], in particular when the attacker probes an unused port (firewall) or uses a probing pattern that deviates from a real usage (IDS/IPS). This proposal does not supersede the use of these defenses but instead delivers an additional layer of security that is comprehensive and quantifiable.

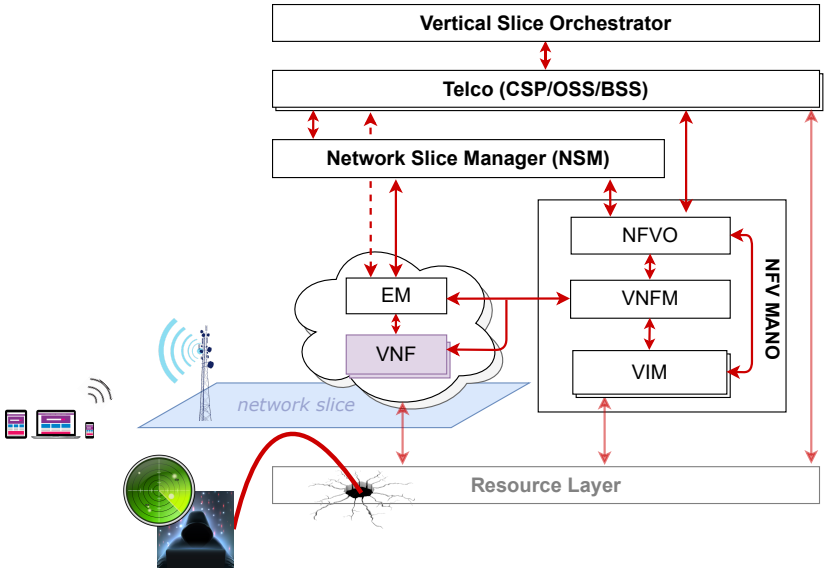


Figure 6.2: Reconnaissance in an NFV powered Network Slice through some other illegitimate access to a management network.

6.3.1 The DynSec approach

Adopting an MTD network parameters mutation approach, and a hard expiration period for the active configuration (*i.e.,* the listening port is only valid within that time window), restricts the attacker to do all probing and the later stages of the attack within that timeframe. Otherwise, the information gathered will be faulty or unactionable. Because of this, the P_{hit} no longer improves reliably at every probe, neither is $P_{hit} = 1$ an immediate consequence of probing all ports, since time is now a factor and ports probed outside of the same "holding still" window does not yield valid information.

This time boundary (T) is a crucial element of the proposal, playing to the fact that the existing definitions of network slicing already include at least one well-defined time boundary: the latency KPI of the slice or, indirectly, the throughput. The maximum amount of probes a single terminal can do over a connection is either $\lfloor \frac{T}{latency} \rfloor$ if done consecutively or, at best, $\lfloor T \times \frac{throughput}{probe\_size} \rfloor$ if done in parallel. α denotes the largest of the two. However, because some configuration spaces are just too small and easy to brute-force even within those constraints (as is the case of the listening port example), the model must allow setting a maximum number of failed tries (k) after which the terminal is flagged as an attacker. Thus, the maximum allowed probes per terminal in each time boundary is $m := min(k,\ \alpha)$. As attackers may also distribute their process across multiple coordinated terminals, it is necessary to count the resulting unique probes against the network slice carried out in each time boundary:

$$u = unique\_probes(m) := \begin{cases} m \times allowed\_terminals & , \quad total\_configs > (m \times allowed\_terminals) \\ total\_configs & , \quad total\_configs \leq (m \times allowed\_terminals) \end{cases}$$

Because loss is perceived differently from gain [129], the proposed model includes the weighting factor for the risk $w \in \mathbb{N}$ that permits penalizing loss of security more harshly. Therefore, we can define the dynamic security ($DynSec$) metric as:

$$DynSec := (P_{miss}(u))^w \times 100 = (1 - P_{hit}(u))^w \times 100$$

When $w = 1$, *DynSec* is a textbook representation of the P_{miss} of the MTD protected function(s) within that exploration space. When $w > 1$, *DynSec* becomes a perceiving metric that emphasizes exponentially the risk of being *hit* .

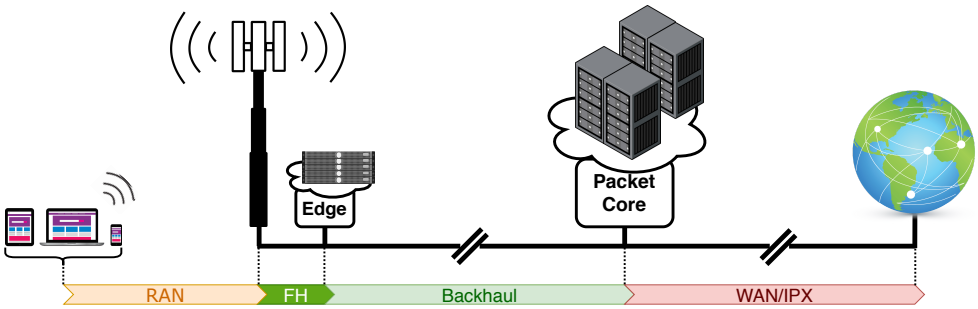


Figure 6.3: Coarse view of the different major network segments in a mobile network

The limiting effect of the time factor can be further enhanced by going more in-depth into how network slices are realized. Figure 6.3 shows a brief break-down of the major network segments that together create the network slice. Four major segments are identified: the Radio Access Network (RAN), the Fronthaul (FH), the Backhaul (BH), and then the Wide Access Network (WAN)/IP Exchange interconnection (IPX). Each of these segments has different intrinsic latencies, either due to the limitation imposed by technology/physical media (*e.g.,* RAN), cable length and speed of light, or even average load. A noteworthy fact is that the network perimeter (RAN and WAN/IPX) usually has larger latency than the fronthaul or backhaul.

The defenders' can exploit this characteristic to their advantage, tightening the mutation period (T) to the functions that only communicate within internal segments. Therefore, an attacker cannot achieve remote connectivity from the higher latency segments to the protected inner-segments function, as long as the MTD mechanism remains unpredictable, or another unprotected component within the lower latency segments is not used to conduct the attack. The use of MTD under this particular scenario will yield security in the same standing as a firewall ($DynSec = 100\%$). In turn, the MTD mechanism will only offer a form of mitigation ($DynSec \in [0\%,\ 100\%[$) when the latency still allows contacting the function.

The *DynSec* metric is a straightforward and comprehensive representation of basic network security, that does not depend on internal knowledge of the function being served, therefore suited to be a feature offered as a service by the slice provider.

6.4 Evaluation

The proposed *DynSec* approach was evaluated using the Monte Carlo method to determine if the modeling matches the outcomes when using an existing MTD mechanism. To construct the experiments, HMAC based [132] listening port mutation mechanism was selected, which takes as the shared secret the concatenation of the tuple $\{secret\_key,\ time\}$. The time factor for these experiments uses Lamport clocks, being the clock synchronization an implicit process upon the number of packets sent/received. Outside of a traffic snooping attack, the attacker uses the best strategy available to enumerate such an MTD protected system, which is a non-repeated random port per probe. Crucially, Figure 6.4 shows how the proposed *DynSec* metric fared with various maximum probe limits (u), that would be set as a function of intrinsic properties of the network slice. On the left side (Figure 6.4-A), it is depicted the expected outputs of the model with the parameters in the test. On the right side (Figure 6.4-B), it is

depicted the experimental results obtained through Monte Carlo experimentation. In this scenario, there were 10000 runs per value, which is sufficient to get a coarser view of the behavior. It can be observed that the experimental results follow the expected outcomes closely, despite the coarser nature of the experiment, therefore validating the correct formulation of the model.

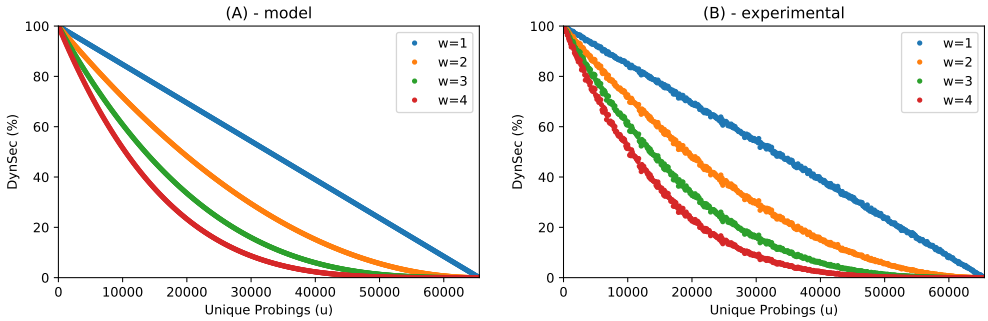


Figure 6.4: Comparing the expected values against the Monte Carlo results across the whole domain

It also evaluated if the model is accurate to select the maximum unique probes ($u = m \times terminals$) so that *DynSec* becomes alike a network SLA. To do so, the experiment was repeated with a higher number of runs (2×10^6 per value) in the typical interval of SLAs (99.95% to 100%). Figure 6.5-A shows the expected u from the model, while Figure 6.5-B shows the experimental results. The modeled u is close to the experimental, but sometimes *DynSec* is slighlty lower than in the model (*e.g.,* when $k = 4$), which is a characteristic of such probabilistic models. Therefore, while the *DynSec* model is representative as a KPI, and workable in a form of SLA, it is not advised to claim stringent assurances of security using just *DynSec*. When looking at such narrow intervals (as in Figure 6.5), the results indicate that the approach is more suited for scenarios where the risk factor (w) is low. This is consistent with the proposal ethos of providing basic security where otherwise no considerations would be made. Networks with a lower number of terminals, as is the case of the management interfaces of the network slice (higher risk factor), could still use the *DynSec* approach effectively as an additional layer of security.

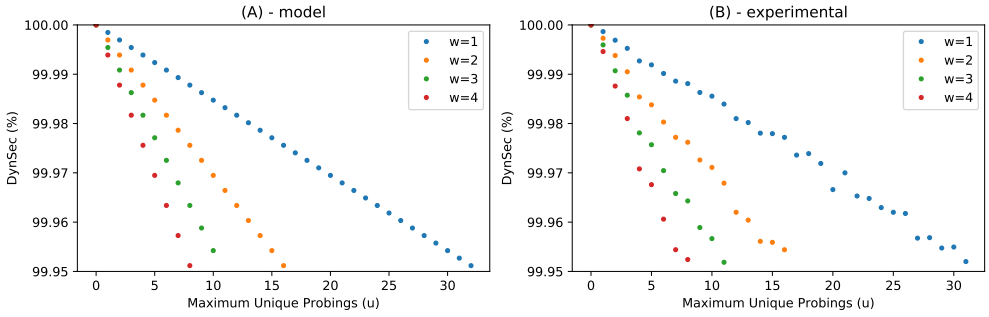


Figure 6.5: Attempting to set the proper u to reach an SLA-alike *DynSec*

6.5 Summary

We have proposed and successfully validated the *DynSec* model that uses MTD mechanisms to create a security KPI for network slicing. The *DynSec* model is a comprehensive representation of basic network security that arises from the fact there is very straightforward discrete math to quantify the probability of hitting a moving target within finite configuration spaces. In this context, hitting the

target means being capable of connecting to the network function(s); therefore, an attacker cannot do reconnaissance or even remotely compromise network functions to which it cannot connect. The model was validated using Monte Carlo experiments, which showed that the model represented correctly the probabilities of each probing attempt hitting a function. The experimentation showed that the *DynSec* model could be used to quantify accurately the basic security provided by the MTD mechanism within the slice, and even create some form of SLA. However, we advise against relying solely on the MTD approach and the *DynSec* metric to make stringent claims of security. The proposed model is not a replacement to other forms of well-established defenses that protect against different aspects of security specific to the use-case, but rather an additional layer of security.

The delimitations of this work present ample opportunities for future work within the orchestration of such MTD mechanisms. For starters, introducing an MTD mechanism will have computational requirements and potential adverse effects over other fundamental KPIs of the slice (*e.g.,* latency), that must be resolved before delivering a given *DynSec* value. Network slices may also exist across administrative domains (interdomain or federation), requiring further study of the challenges these may present.

CHAPTER 7

Integration into research testbeds

This chapter documents this thesis's dynamic security mechanisms integration efforts into real-world use cases, such as the 5Growth pilots, and discusses their contribution to those pilots.

7.1 The H2020 5Growth Stack

The prototype for the inter-domain E2E vertical slice is implemented according to Communication Service Management Function (CSMF)-to-Network Slice Management Function (NSMF) multi-domain communication model. The CSMF-C of the 5Growth Vertical Slicer (5Gr-VS) is extended with procedures to request multiple network slices to multiple NSMFs, either local (i.e., managed by the 5Gr-VS) or remote ones. The interaction with the NSMFs is handled through a software component that offers the CSMF-C a unified interface inspired by the 3GPP standards related to the NSMF APIs (3GPP TS 28.531 [133]) and the network slice resource model (3GPP TS 28.541 [134]). Internally, the NSMF-specific drivers translate the CSMF-C's requests in the format supported in each external domain and either map the respective NSTs or decompose the slice request into the respective network services. ETSI OSM is used as each domain's service orchestrator. Therefore, an OSM adapter translates the messages for instantiation, termination, and queries of network slices from/to TS 28.531 format and the decomposition to suitable network services controllable through the OSM REST API.

Because these inter-domain communications go through a public network, we must make additional security considerations to ensure proper operation in the above management interfaces (i.e., integrity, confidentiality, and availability). Furthermore, the E2E inter-domain forwarding plane used by the vertical services' must also be protected. Therefore, we have previously researched secure and reliable network slicing [135] and adopted three approaches to tackle some of the challenges: (1) we have used an MTD mechanism [136] to protect the inter-domain management interfaces against unknown types of attacks (e.g., 0-days); (2) we assure reliability in the forwarding-plane through performance isolation and QoS using SDN and a mixed Openflow/P4 data-path; and, (3) we have an AI/ML-assisted anomaly detection mechanism.

However, our previous forwarding plane assurances focused chiefly on performance isolation and required controlling all network switches in that path, which is an extreme assumption within this PNI-NPN scenario. Thus, we complement our previous research by introducing secure tunnels between the domains to allow for greater flexibility. We may have to sacrifice some of the performance guarantees but will still retain the integrity and confidentiality in the public transport network. Our design

contemplates separate secure tunnels for the management and each of the vertical service's forwarding planes. We take a deeper look into the latter later in the chapter.

Figure 7.1 shows the infrastructure used for the evaluation scenario of our PoC. Each of the rack-mountable server units represents a Dell PowerEdge R430 server equipped with 2xIntel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz. The most significant difference between the different servers is the amount of RAM installed and the type of storage used. The compute cluster of Domain A's Openstack uses a Ceph based storage that spans across the local disks installed in each compute node. In turn, the compute cluster of Domain B's Openstack and the Proxmox cluster use the same centralized Dell SCv2020 DC storage unit with 24x 1.2TB SAS 10k 12Gbps 2.5" harddrives in RAID 6. The Openstack controller of the Domain A is containerized and virtualized within that hardware node, having 6GB of RAM allocated. Domain's B controller runs directly on the hardware and has 256GB of RAM installed to perform a large block chache of the centralized storage. Each of the VMs in this deployment was dimensioned for the PoC, using monitoring data gathered during early runs. The portal and 5Gr-VS VMs each had 2 vCPUs and 8GB of RAM. The OSM VMs were given up-to 8 vCPUs and 32GB of RAM depending on the current usage. Each of the service VMs instantiated in the respective domain's Openstack had 4 vCPUs and 4GB of RAM.

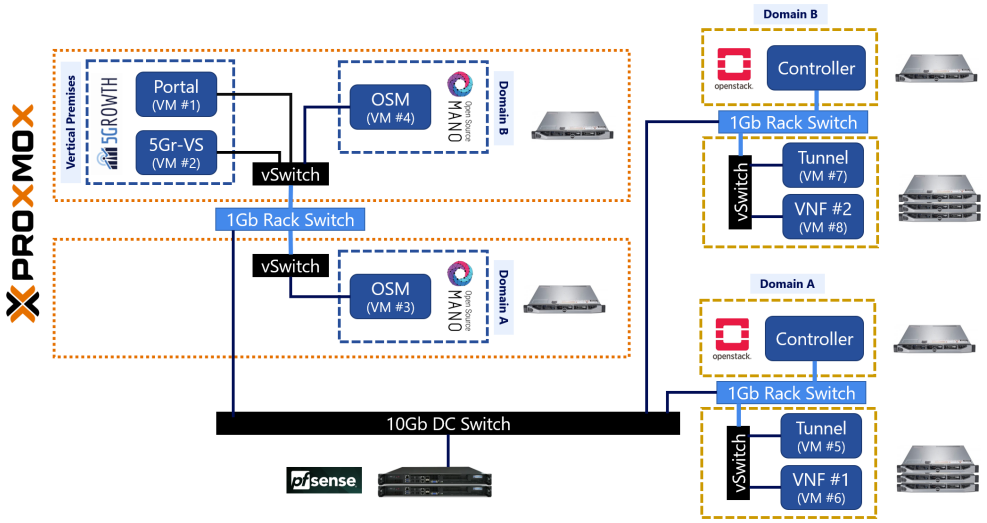


Figure 7.1: A detailed overview of the infrastructure used for the evaluation scenario of PoC

The simulated IXP/ISP environment used for the interdomain communications consists of different 1 Gbps copper networks within a data center. Each network is local to the switch of the respective compute cluster that runs the 5Growth software and each of the VIM domains. Each of the domains under evaluation is placed in physically distinct compute nodes. The local networks are then routed centrally by the data center's firewall, a redundant pfSense system with 10 Gbps fiber links to the switches. The dataplane of the E2E slice is achieved using secure tunnels (Wireguard) that carry the packets across domains, being those tunnels overlaid atop the simulated IXP/ISP environment. We will provide further details about the tunneling solution in the respective evaluation subsection.

7.1.1 Results of Network Slice Level Solution (PoC)

Figure 7.2 depicts the evaluation scenario for the Network Slice Level Solution. It implements the network slice multi-domain solution, where the vertical service under evaluation consists of a simple service composed of two instances of a probing VNF. The probing VNF aggregates local sensorial

data (e.g., train and track informational telemetry) in closer proximity with the provider where it is instantiated. The probing VNF collects forwarding-plane performance data (throughput and latency) and evaluates if the E2E slice complies with the KPIs specification. The instantiation of the E2E vertical slice requires an additional VNF to handle the secure tunneling (i.e., Wireguard) that interconnects the two sub-slices. The 5Gr-VS instantiates the secure tunnel VNF, an internal component of the 5Growth inter-domain solution, and seamlessly manages it without the vertical's intervention. The experiment results cover the most significant interactions required to instantiate the inter-domain slice across domains A and B, profiling the various steps' delays. Each of these steps is represented in Figure 7.2 and further described as follows:

- **Step 0:** A vertical's human operator enters the pilot's 5Gr-VS Portal. He will then select the vertical service that will result in a multi-domain E2E vertical slice with the two probing VNF instances and request its instantiation. The delays resulting from the human operator are omitted. In doing so, it is just measured the loading time required to show the instantiation page, fully rendered with all the dynamically acquired information about available templates, resources, and other system states. The page load took on average 238.32 ± 5.96 ms.
- **Step 1:** The pilot's 5Gr-VS Portal maps the vertical service request into the appropriate templates/descriptors and translates the user request into a compliant request to the 5Gr-VS CSMF. This step considers the elapsed time between pressing the action button in the instantiation form and triggering action in the 5Gr-VS CSMF. This step takes an average of 9.95 ± 1.54 ms.
- **Step 2:** The 5Gr-VS CSMF decomposes the E2E vertical slice into separate sub-slices. It requests their provisioning and management to the respective domain's 5Gr-VS NSMF, parallelizing the requests to separate domains (e.g., *a* or *b* in Figure 7.2). The 5Gr-VS CSMF also requests the instantiation of the secure tunnel VNF in each sub-slice and instructs the 5Gr-VS NSMF that the tunnel VNF must be configured through the respective primitive with endpoint information that arises from the resolved addresses after VDU instantiation. The step takes an average of 216.69 ± 26.67 ms.
- **Step 3:** The 5Gr-VS NSMF translates the sub-slice request into the appropriate artifacts of the domain's NFV-NS (i.e., using network slice templates or composition of network service descriptors). The 5Gr-VS NSMF will then request, in parallel, that each SO instantiates all components required to build the sub-slice. The NFV-NS Life-Cycle Management (LCM) is now delegated to the respective SO. The parallel requests took an average of 4.09 ± 0.70 ms in domain A and 3.50 ± 0.52 ms in domain B.
- **Step 4:** The SO determines which VIM is more suitable to fulfill the request and starts the LCM of the requested components, deploying the service-specific managers and support for service primitives (if the component requires them). In particular, the secure tunnel VNF crucial for the inter-domain connectivity requires a mix of day-1 and day-2 primitives to configure the tunnel endpoints with the resolved VDU information. Step 4 and 5 take 62.90 ± 6.74 s in domain A and 50.94 ± 8.60 s in domain B.
- **Step 5:** The VIM receives the instantiation requests and delivers the virtual resources from its shared resource pool if the resources to fulfill that request are available within the set constraints. Upon instantiation of each VDU, the VIM makes available the resolved endpoint address information. Steps 4 and 5 combined take 62.90 ± 6.74 s in domain A and 50.94 ± 8.60 s in domain B.
- **Step 6:** Upon delivery of the virtual resource, the LCM within the NFV-NS will start provisioning the slice/network services accordingly to the vertical requirements and the computed

configurations during the network slice decomposition process of the upper layers of the stack. This facility will also enable the on-demand configuration of the vertical's services if those primitives exist. The step takes 421.93 ± 67.26 s in domain A and 160.39 ± 13.73 s in domain B.

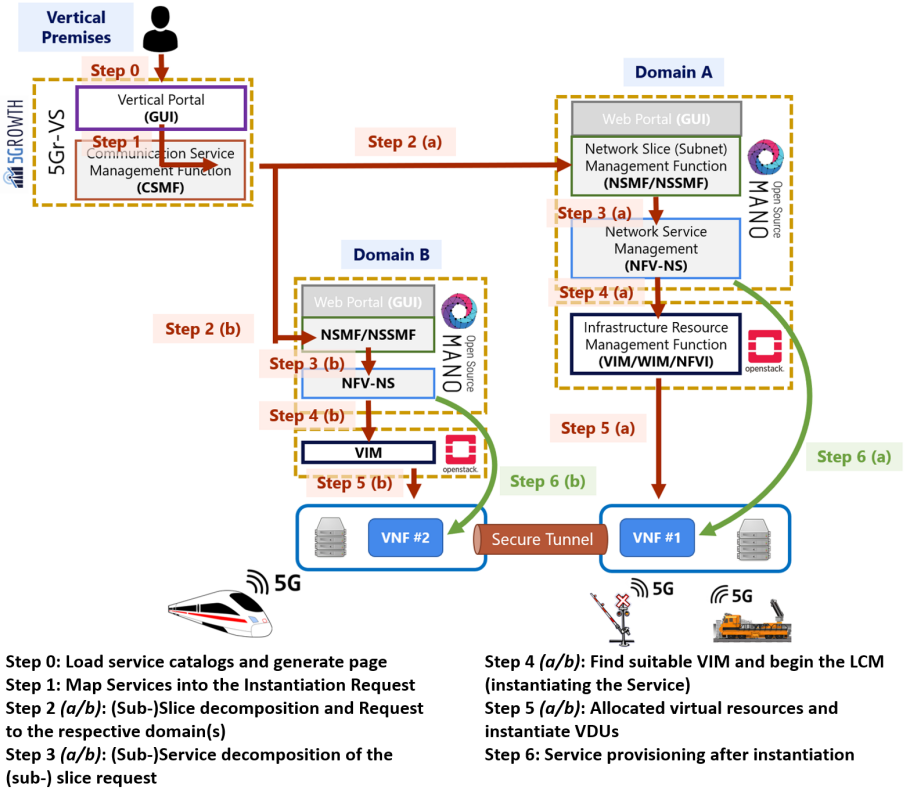


Figure 7.2: Evaluation Scenario of PoC comprising the Integration of 5Growth platform and Two OSM-based Domains (PNI-NPN with full sharing deployment)

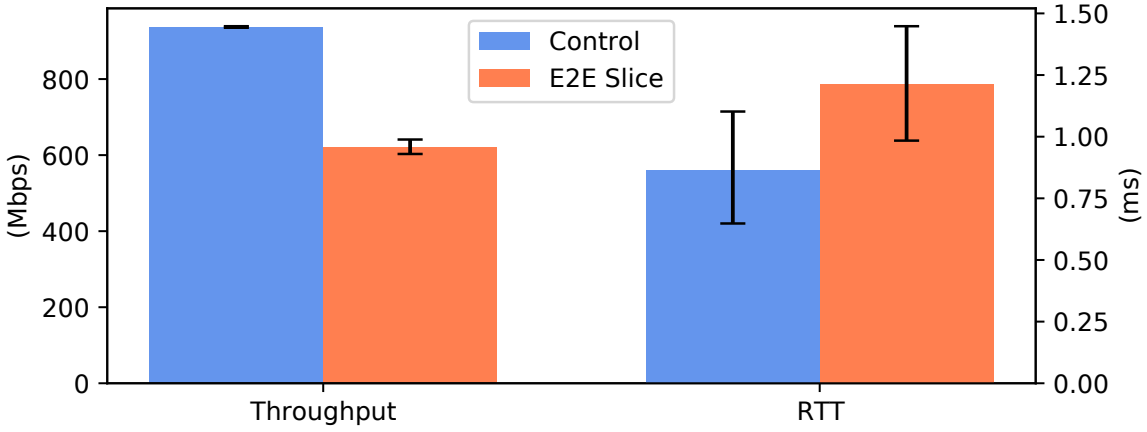


Figure 7.3: Impact on E2E Communication (PoC)

After evaluating the control overheads caused by the inter-domain solution during the vertical service instantiation, the behavior in the slice's forwarding-plane is evaluated. This includes an assessment on if the inter-domain solution can comply with the slice KPIs and determine the interconnection overheads of stitching the two sub-slices into a single E2E vertical slice. Two crucial KPIs are selected for this evaluation: throughput and latency. A control baseline is established in order to show the

network performance between the two domains, which is measured using a E2E slice between the two vertical VNF instances, placed each in a separate domain. The measured forwarding plane performance is shown in Figure 7.3, highlighting that the limiting factor is the secure tunnel that inter-connects the different domains (i.e., a Wireguard tunnel). That tunnel accounts for a drop of nearly 30% in the measured throughput, going from $\approx$ 936 Mbps of the direct connection to $\approx$ 620 Mbps of the vertical slice. Despite the secure tunnel's expressive overhead, this loss must be critically evaluated in the context of the benefits to the E2E network slice and the compliance with the KPIs. The throughput values through the tunnel are reliable, much like the domains' direct interconnection values. Therefore, the inter-domain solution crucially delivers its benefits while still obeying the high throughput KPI (up to the $\approx$ 620 Mbps of $\approx$ 1 Gbps) as long as the domain's underlying interconnection remains compliant with its SLA. As for the measured RTT, because the direct connection already had such a low baseline ($\approx$ 0.86 ms), the E2E slice results in an increase of nearly 40% of KPI (to $\approx$ 1.21 ms). Despite the large percentage, the absolute value does not invalidate its usage for most vertical applications. Therefore, the inter-domain solution obeys the low latency KPI (up to $\approx$ 1.21 ms) as long as the underlying connection between the domains remains compliant with its SLA.

7.2 Management Interfaces Defense

The 5Growth stack is designed to span across different administrative domains connected through a network. Crucially, sensitive communications such as the LCM operations over the vertical services controlled by our stack may go through untrustworthy public networks. Figure 7.4 shows an overview of the different domains used in our PoC evaluation, depicting the running 5Growth Aveiro pilot site. The site has three main stakeholders: the vertical customer, the ICT-17 infrastructure of 5G-VINNI, and the other domain (a private cloud). In red, we have the critical trust boundaries where the sensitive management communications go through untrusted networks. We also show the existing trust boundaries within each administrative domain (for internal management) in gray. We expressly trust the pre-vetted domains that participate in the 5Growth stack but not the public networks used.

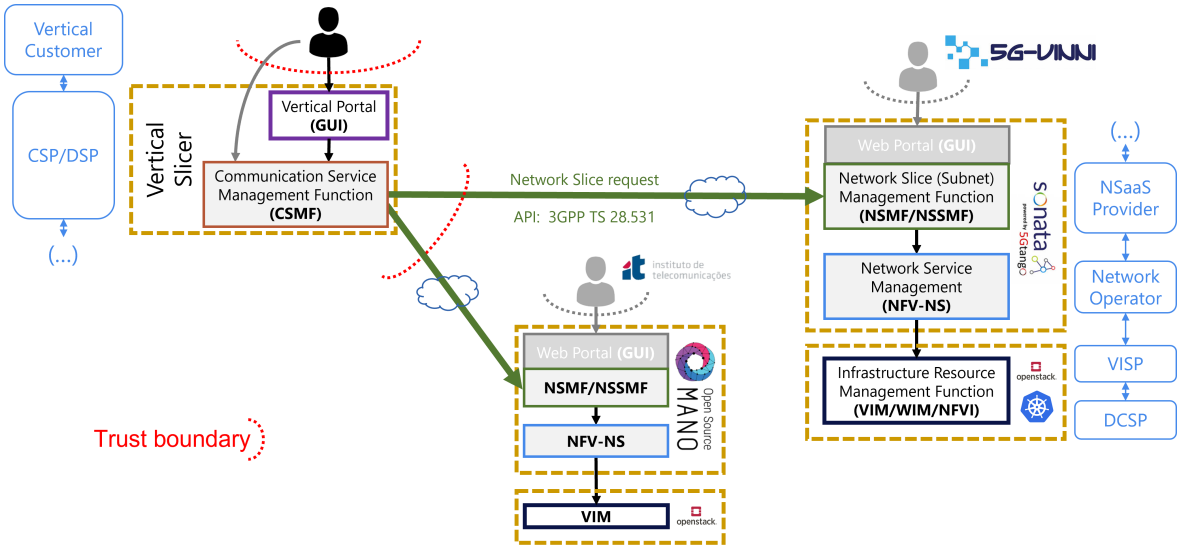


Figure 7.4: 5Growth Management

Protecting the 5Growth stack against attacks originating from the public networks requires a holistic approach that addresses the threat model and assures the CIA triad. The threat model can enumerate

the known attacks. Those can be addressed with common security control approaches, such as adequate authentication, access control, and correct use of cryptography to assure data confidentiality and integrity. In turn, unknown attacks may explore other vulnerabilities outside that scope of the 5Growth stack, such as the lower layers of the operating system's network stack or the programming language frameworks where the standard security controls will be implemented. Thus, unknown attacks are risks that cannot be fully resolved in the threat model. Our research focuses on these unknown attack types, specifically those that can originate from the public networks. We will model this perimeter breach attack process using the CKC [24].

The CKC consists of 7 sequential stages. During stage *(1) reconnaissance*, the attacker enumerates the endpoints, finds its targets, and discovers their characteristics. After acquiring sufficient information during the reconnaissance, the attacker starts the *(2) weaponization* stage. The service characteristics are used to identify a vulnerability and then devise a tailored payload to exploit that vulnerability. Once the payload is built, the attacker enters the *(3) delivery* stage, where that payload must be transferred into the target. After successfully delivering the payload, the attacker can enter the *(4) exploitation* stage where that payload is executed in the target system. That execution gives a foothold to the attacker that allows the *(5) installation* of a more reliable backdoor communication channel. After establishing the more reliable channel, the attacker can embed the victim machine into a *(6) C2* such as a botnet system. Lastly, the attacker can now perform *(7)* actions upon his objectives.

The communications going through the untrusted network are client-server interactions with a single 3GPP TS 28.531 compliant API. Therefore, the interdomain communications' attack surface can be reduced to a single server socket in each domain that must expose that API (i.e., the Network Slice as a Service (NSaaS) Providers). The 5Growth stack already minimizes the attack surface to its minimum, a single API socket that must be exposed to authorized parties. Furthermore, adequate controls are in place to defend against the known attack types. Nevertheless, the attack surface can never be zero (i.e., we need to expose that socket to deliver functionality), and the unknown attack types may still subvert our security controls. Our main contribution has this fallibility in mind and thus introduces complementary defense mechanisms that increase the resilience against an unknown attack. We use MTD to create a seamless layer of authentication that moves the attack surface unpredictably to the attacker, thus disrupting the attack process at the early stages of the CKC. The MTD works at the routing/forwarding level, and it is not an endpoint terminating the connections.

Within the unknown attack types, we have two main threat scopes: attacks carried out by actors that do not have inside knowledge and the attacks performed by advanced threats that can acquire inside information through other intelligence sources. Actors without inside knowledge rely heavily on the reconnaissance processes to acquire actionable intelligence. When the attackers already have some inside knowledge of the system and its vulnerabilities, we must disrupt the malicious payload delivery, thus stopping our stack's further exploitation through this vector. Our attack surface is characterized by three network parameters within TCP/IP network headers: the transport protocol (e.g., TCP), the listening IP address of the host, and the listening port. Delivering a malicious payload that can inflict damages to our stack relies firstly on successfully filling the correct network headers data that allows reaching the service. Our approach builds upon the DynSec straightforward optimization with probabilistic defense that used MTD to set network slicing security as a KPI (Chapter 6), delivering valuable data about its performance in a PoC system. Our MTD mechanism relies on HMAC [137] to produce reliable mutations that can only be reversed by the authorized parties (i.e., those who hold the shared secret). In effect, the mutations are an additional seamless network-layer authentication mechanism that stops unauthorized communications from being terminated in a socket. Thus, we

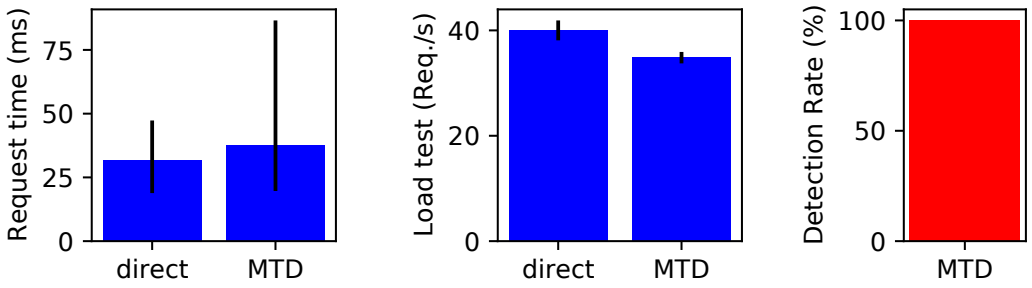


Figure 7.5: Interdomain VS-Sonata LCM Requests Performance

defend against unknown attacks by preventing the unauthenticated processing of the malicious payload.

The evaluation was done through a PoC using the Aveiro pilot's facilities. We started by determining the critical mutation period for our mechanism. We have measured the typical SONATA API request time for the LCM interactions and determined that (on average) the response took a little over $\approx$ 30 ms. Therefore, we have set the MTD mechanism to perform a mutation every 30 ms. The SONATA API performance values shown in Figure 7.5 are within the 95% confidence interval. The bars show the calculated average, and the wishers show the distance to the min-max. The response time measurement was conducted in 500K runs for each of the two cases. We used siege to carry out the load test. The load test values were gathered in 100 runs, each run having 10 workers that will perform 100 requests in parallel, making a total of 100K datapoints. Additionally, we experimented with different parallelism levels, finding that the Sonata performance results were similar despite the changed parameters. We have determined that the MTD mechanism has a measurable impact on the SONATA API's performance, making the average request time $\approx$ 19.53% slower, and reduced the number of requests attainable per second by $\approx$ 15.21%. Despite the initial impressions caused by those numbers, the impact over the average request time ($\approx$ +6.16 ms) is not significant to the management processes done through that interface. Furthermore, while there is an impact to the maximum requests per second attainable when stress-testing the SONATA API (≈ -5.29 req./s), that is hardly a typical scenario in the 5Growth stack. We found that the MTD performance is well within the pilots' requirements.

In turn, the security benefits of the MTD mechanism were well noticeable. We have placed an attacker that attempts to exploit an information disclosure vulnerability in the SONATA API. All authentication mechanisms were disabled, leaving just the MTD solution as the only line of defense. We have carried out 500K runs of an exploit attempt of the SONATA API using the best strategy available (a random port). In Figure 7.5, we show that the detection rate was $\approx$ 99.9958%. In the few instances where the attacker evaded detection, we did not record any exploitation: the connection reseted midway because the mutation period was so fast (30 ms) and below the average request time ($\approx$ 37.67 ms with MTD).

7.3 Dataplane Security

After evaluating the management plane security, we move to the Network Slice's dataplane. Because the 5Growth project contemplates interdomain network slices, it is assumed there is interconnecting overlay tunnel across the domains.

Figure 7.6 illustrates this concept, a MANO and Infrastructure view with two Networks Domains (A and B). Each interdomain Network Slice comprises N separate Network Slice Subnets, one for

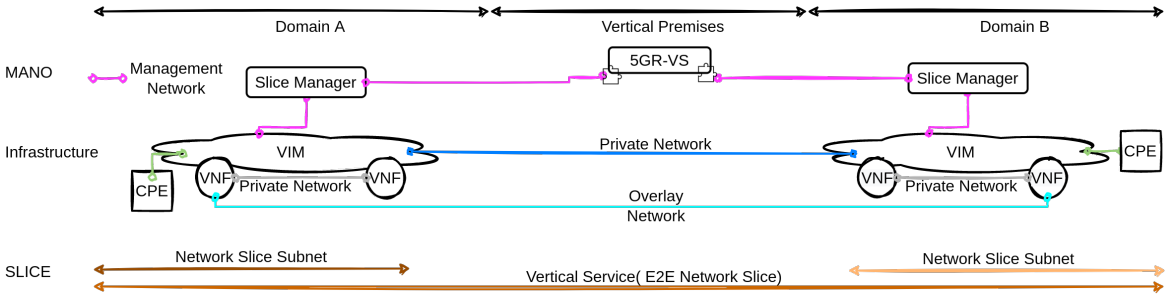


Figure 7.6: Interdomain Network Slicing concept

each Network Domain. The Network Slice Subnets are created using single domain Network Slices, leveraging NSaaS. To ensure this, each Network Operator (NO) needs support managing Network Slices through their respective slice manager.

Focusing on the Infrastructure view, we need a tunnel to ensure the stitching of the network slice subnets while still ensuring the fulfillment of the KPIs defined by the Vertical. In our first approach, we have used Wireguard VPN to guarantee secure and reliable communication between domains. The Wireguard VPN minimizes the attack surface (*i.e.,* lines of code); to this date, Wireguard has no significant known vulnerability. Nevertheless, an undiscovered vulnerability may still exist in that code path leading to a 0-day. Because of that, our solution employs an MTD approach that adds a new layer of defense (*i.e.,* port mutation). In particular, atop the Wireguard protection, the MTD mechanism will change the tunnel's UDP ports accordingly to the respective peer's TOTP authentication. The attacker must first discover the right OTPs that determine the currently used UDP ports. Otherwise, outside of sheer luck, it is nearly impossible to deliver the malicious payload or proceed with further enumeration without being detected. The mechanism herein deployed and evaluated is the TOTP-MTD [136] approach described in Chapter 4.

7.3.1 Evaluation

The test environment uses two Virtual Infrastructure Managers (VIMs) connected through a private dedicated link (1 Gbps). Upon creating the End-to-End (E2E) Network Slice, in each VIM, there is a VM dedicated to create an overlay network (*i.e.,* tunnel) between the two domains VIMs ensuring an E2E private connection between the Customer Premises Equipments (CPEs). Furthermore, the MTD function changes the ports being used by the tunnel service, allowing for communication protected against enumeration and exploit attempts. Wireguard will encrypt the traffic, authenticate its peers, and perform integrity control using a shared secret to communicate with each domain. To evaluate the interdomain solution's effectiveness in this thesis, we eliminated the tunnel's cryptographic overheads and used the MTD function plus VXLAN to defend against this malicious action. VXLAN is also often used in the industry and benefits from various performant implementations.

We designed four scenarios to evaluate the correct operation of the Network Slice Subnets stitching over multiple administrative domains, measure the impact, and evaluate two MTD security (determined by the mutation speed). We now explain in summary the analyzed scenarios:

- N- The traffic of the CPEs communicates without any rerouting to the VNFs (control scenario)
- T_VX- The traffic of the CPEs goes to the VMs that support the VXLAN overlay network
- T_VX+MTD(100)-The traffic of the overlay network is rerouted to the MTD function with configuration for a TOTP change every 100ms
- T_VX+MTD(30)-The traffic of the overlay network is rerouted to the MTD function with configuration for a TOTP change every 30ms

When the MTD function is enabled and chained into the service, seamless network authentication happens at every packet using the exposed UDP/TCP service ports (i.e., the VXLAN tunnel UDP ports in the scenario). When that per-packet authentication fails, the respective packet drops, and the thread monitoring system receives an event for further analysis and decision. The MTD mechanism relies on TOTP authentication, which utilizes the clock to calculate each OTP. Therefore, events such as network jitter or congestion may cause a failed authentication alert without adversarial action. The following section critically evaluates these effects in inter-domain communications.

In terms of performance evaluation, the targeted indicators include throughput, retransmission rate, and end-to-end latency, The tests employed mainly the Iperf3 [1] and the Linux Ping tools. The VNFs are instantiated in VMs that run the Ubuntu 18.04 LTS Cloud Image [2] and the tools available through the package manager. The experiment used Iperf3's default configuration. Therefore, the iperf3 tests use TCP and the *cubic* congestion protocol, the Linux distribution's default.

This section critically analyzes the results of the different tests devised to evaluate VXLAN for stitching two domains.

We have measured the throughput in all four scenarios and verified a $\approx 10\%$ decrease attributable to the use of the VXLAN tunnel VNF compared to the control scenario. We expected less usable payload data due to the additional tunnel header in each packet. Figure 7.7 shows the measured throughput. Unlike the VXLAN tunnel overheads attributable to added bytes for the tunnel header, the MTD function does not add any bytes to the packet (in the data plane). Focusing on the throughput, adding the MTD function caused a further $\approx 8\%$ decrease (on average) with higher variability. The reduction is likely due to missed OTPs causing retransmissions. When port changes are more frequent, we have more OpenFlow commands, and the performance also has a higher variability. Because the SDN controller and the software switch communicate within the same VM, these control messages do not influence our results.

Because reliability is a vital network slice KPI, we have also measured the retransmission rate during the Iperf3 high line load tests. As shown in Figure 7.8, we have verified that the retransmission rate slightly decreases when using the VXLAN tunnel. We attribute this result to the TCP and Iperf3 inner-workings detecting the maximum line rate within the tunnel. We have also verified that the retransmissions increase when the MTD functions are enabled, and the results vary. Similarly to the throughput evaluation, we attribute these results to transient effects caused by frequently and rapidly changing the OpenFlow rules in an ongoing flow within the software switch.

The MTD function did not cause any dropped packets due to failed OTPs when mutating at a 100 ms interval. However, the 30 ms mutation interval yielded a lowered packet rate of $0.0424 \pm 0.0258\%$. These failed authentications get enforced as false positives in the data plane; however, the threat analysis system can eliminate these false positives for the alarm generation described in the TOTP-MTD evaluation. We have used that approach and experimentally reached a 0 % false-positive alarm rate. The measured false-negative rate was meager ($\approx 0.0042\%$).

The RTT evaluation showed significant differences between making the measurements with an unsaturated line (*i.e.,* the Ping command) or a fully saturated link with Iperf3 (as shown in Figure 7.9). When the line is unsaturated, using the VXLAN VNF adds a 0.75ms delay to the RTT. We expected this result because the tunnel VNF adds one more hop to the chain (*i.e.,* one extra VM to traverse). Similarly, introducing the MTD function adds another extra hop and thus an extra 0.75ms. When the Iperf3 test loads the line, the RTT is nearly double compared to the baseline (6ms vs. 12ms). We have

[1] `https://iperf.fr/`
[2] `https://cloud-images.ubuntu.com/bionic/`

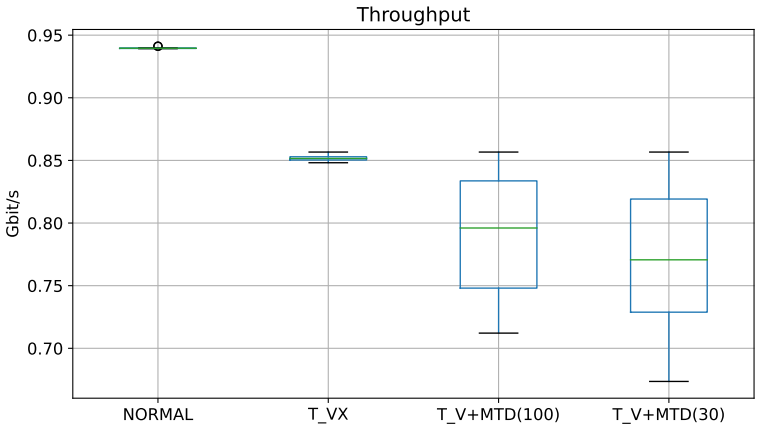


Figure 7.7: Available Throughput (iperf)

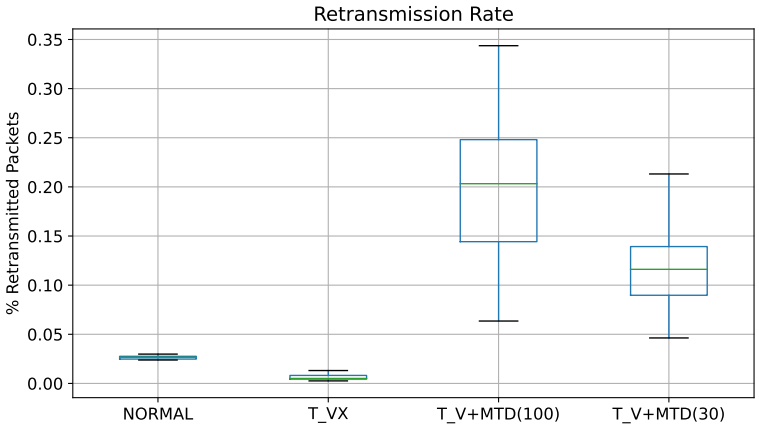


Figure 7.8: Retransmission Rate (iperf)

also observed that the rapid use of OpenFlow commands to the software switch enforcing the OTPs causes additional jitter. Nevertheless, the interdomain solution is working predictably and in line with the expected effects of service function chaining. Furthermore, the experimental evaluation shows that the interdomain solution is workable and feasible for deployment.

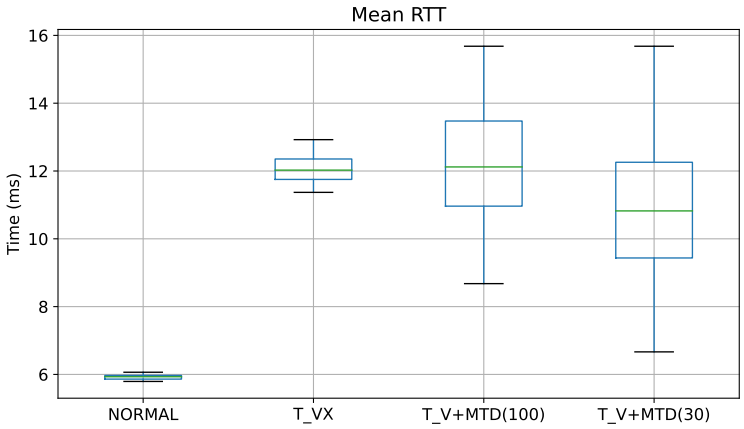


Figure 7.9: Mean RTT with TCP (iperf)

7.4 Summary

This chapter describes the integration into research testbeds and critically evaluates the use of this thesis's MTD mechanism within the H2020 5Growth stack. The results show that the mechanism is practicable for defending both management interfaces and the dataplane overlay. However, further research is still necessary to ensure the interdomain overlay dataplane can deliver the selected Network Slice's KPIs. That research relates mainly to the system aspects of the 5Growth stack, not the thesis's direct contributions. Nevertheless, in its present form, the MTD mechanism proved to be usable in the interdomain scenario.

CHAPTER 8

Conclusion

This chapter summarizes this thesis's primary findings and contributions. We critically discuss the achievements in light of the proposal and the integration made with real-world use-cases. Then, we conclude the thesis with future work and other enhancements that were not possible within this thesis time plan.

8.1 Summary

Chapter 3 identified the related works and set the working framework for this thesis' contributions. Network slicing systems use softwarized and virtualized networks, usually built atop ETSI-NFV compliant systems in the Telecom world. Therefore, creating a practical MTD approach was fundamental and addressed in the next chapter (Chapter 4). Our TOTP-MTD solution successfully delivers a suitable port-mutation mechanism for deployment in Edge, 5G, and other Telco systems. The design and technical approach solved the decisive challenges hindering the existing port-mutation solutions: maintaining mutation synchronism while delivering the necessary flexibility and soundness.

TOTP eliminated the need for an exclusive peer synchronization protocol, avoiding those overheads and other solution scalability issues. Our experimental results demonstrate that the mechanism works with excellent forwarding-plane performance and can tolerate packet loss without losing synchronization permanently. Crucially, the mechanism can sustain quick mutations (in the order of milliseconds) even with multiple peers and unique keys per peer.

Having NFV at the core of its design and packaging the solution as a VNF allows great flexibility and integrates nicely with the working framework. For instance, we could leverage the NFV-MANO interfaces for necessary actions such as adding/removing peers, configuring mutation parameters per peer, and crucially enabling horizontal scalability. We have successfully validated the horizontal scalability using PBR and shown the effects of adding peers to a running instance (within its entire operating range). Forwarding-plane performance and OTP generation load remained mostly unaltered while the enforcement's CPU load rose to a plateau. Furthermore, our design allows offloading the demanding enforcement to regular SDN switches.

TOTP-MTD reuses the proofs and protocols of an already vetted 2FA solution, not a custom function that may require recertification for industrial compliance. The time-based approach puts a hard limit on the window of opportunity in traffic interception/MitM attacks. We have experimentally validated that the approach stops unauthorized access and detects threats to the system (*e.g.,* probing

attempts). The solution design sets distinct thresholds for enforcement and alarming. We enforced stricter limits in the forwarding plane (to stop adversarial action) and then devised a false-positives elimination method in the alarming phase.

We have achieved excellent all-around performance (latency, throughput, OTP enforcement, and threat detection) despite our PoC implementation's straightforwardness. However, there is still room for improvement (outlined in the future work subsection that follows).

We have designed an enhanced response and information gathering mechanism leveraging the detection capabilities of the MTD mechanism (acting as the IDS). In Chapter 5, we have effectively used the `TCP_REPAIR` socket options built into the Linux Kernel to devise a proxy solution that can receive an active TCP connection (*e.g.,* an attack) and seamlessly continue that session with a new endpoint (*e.g.,* the decoy). Our solution's performance was near the baseline of a regular socket, with the latency nearly the same. At the same time, the average throughput was reduced by less than 5% of the regular socket. That was a great result, especially when compared to the alternative approaches of `NFQUEUE` (honeybrid) and the SDN Controller as the packet mangling frontend (one of the operating modes of HoneyDOC), which affected the latency and the throughput very significantly (different orders of magnitude in throughput). This proxy function allows raising the utility of our MTD function as a detection mechanism. It also allows adding extra capabilities to an existing IDS/IPS system.

The `TCP_REPAIR` Proxy handles the live TCP session piping to the new endpoint, not the handover required to steer the traffic. Therefore, we have devised a handover sequence that leverages the SDN controller and integrates it nicely into the Slice Manager to solve this issue. We have experimentally validated that our handover approach worked correctly and gathered all performance data from the `TCP_REPAIR` Proxy already using this handover solution. We leverage standard features of the OpenFlow 1.5 protocol to match specific TCP flags to gather (via `PACKET_IN`) the critical three-way handshake data (*e.g.,* negotiated Maximum Segment Size, window scale factor, and sequence numbers) and keep track of the active TCP connections. Using a previous OpenFlow version will come at an efficiency loss, as gathering this data would require sending all TCP packets to the controller.

We have qualitatively assessed the complexity of this solution for use within network slicing systems and concluded that the integration footprint would not be of higher complexity than the already existing modules.

Researching and providing ways for minimizing overheads (*i.e.,* computational resources optimization) and defining a security KPI is the first step towards smarter orchestration in an ETSI-NFV system. In Chapter 6, we have proposed and successfully validated the *DynSec* model that uses MTD mechanisms to create a security KPI for network slicing. The *DynSec* model is a comprehensive representation of basic network security that arises from the fact that there is very straightforward discrete math to quantify the probability of hitting a moving target within finite configuration spaces. In this context, hitting the target means being capable of connecting to the network function(s); therefore, an attacker cannot do reconnaissance or even remotely compromise network functions to which it cannot connect. The model was validated using Monte Carlo experiments, which showed that the model correctly represented the probabilities of each probing attempt hitting a function. The experimentation showed that the *DynSec* model could accurately quantify the basic security provided by the MTD mechanism within the slice, and even create some form of SLAs. However, we advise against relying solely on the MTD approach and the *DynSec* metric to make stringent claims of security. The proposed model is not a replacement for other forms of well-established defenses that protect against different security aspects specific to the use case, but rather an additional layer of security.

Finally, the integration with a vertical slicing system and realistic use-cases is fundamental for validating the proposed solutions and acquiring further data on how to improve the approaches for real-world deployment. In Chapter 7 we have shown that the MTD approach is practical and capable of defending the management interfaces of that system and the interdomain dataplane overlay.

We will proceed to evaluate this thesis contributions in the following subsection critically.

8.2 Discussion

In order to **dynamically change security in a softwarized and virtualized infrastructure without disrupting existing services or functionality** (Q1), thus achieving G1, G2, and R1, service continuity strategies will be researched as a means to allow MTD solutions to be deployed on-the-fly. Such continuity strategies should contemplate different levels of the OSI stack, as required for each particular service. For instance, intuitively, an application proxy approach would be the easiest way to achieve continuity. However, such an approach would come at a loss of generality (*i.e.,* it needs a different proxy for each application). By holistically providing ways to resume operation at lower levels of the OSI stack (*e.g.,* TCP sessions), we would alleviate the efforts required to maintain service continuity, enabling existing management and orchestration methods for dynamic changes in defense mechanisms and configuration. Therefore, the second mechanism described in Chapter 5 directly addresses these points.

Assessing if **concurrent defenses deployed dynamically and simultaneously will improve security over a single defense** (Q2) is directly related to the question: **How does concurrent defense mechanisms' effectiveness scale when related to the non-concurrent instantiations of each mechanism** (Q3). Both Q2 and Q3 build upon the answer to Q1 and directly relate to the early research of *Connel et al.* [81] regarding concurrent defenses. We have shown that having two separate mechanisms and dynamic changes through orchestration (Chapter 7) does not invalidate previously observed benefits of individual defenses. However, as already noted by *Connel et al.* [81], the existence of R1 will contribute to the missing research of how different kinds of concurrent defenses interact at different stages of their life-cycle.

Determining if the security gain justifies the cost in overheads was fundamental to the cost-effectiveness objectives of our proposal (G3, G5, and R2). In Chapter 6, we introduced an optimization framework called DynSec. The DynSec approach also allows **delivering the right kind of security at every moment** (R5), being the definition of security suited to the particulars of that service and in that instantiation.

Making the unpredictable MTD movements easily traceable and logically reversible to the defender is a requirement to **retain the defender's home advantage while making the attacker fight a new battle at each new attempt** (G6). In Chapter 4, we devised a mechanism that takes the best of existing 2FA practices to deliver a port-mutation MTD function that makes reconnaissance and delivery much harder. The mechanism contributes to the field with a more suitable solution to the real-world requirements of network operators such as in 5G.

Chapter 7 shows **how distributed locations, multi-tenancy, multi-ownership, and usage billing impact our dynamic security mechanisms** (Q4) and establish effective methods that are aware of these characteristics (G4, R4) further contribute to solving the challenges of newer generation networks, such as 5G, especially in the context of the H2020 5Growth pilots.

Finally, the contributions made to the IEEE P1912 (R5) and the assistance given to master student's thesis (R6) address the final points in the thesis proposal.

8.3 Future Work

The already identified delimitations of this work allow for ample opportunities for future work. The TOTP-MTD mechanism relies on world clock synchronization, predictable network delays, and an adjustable window of still valid OTPs. Predicting those network delays, adjusting the window, or predicting misbehaving clocks is crucial for the correct operation under tighter (*i.e.,* faster) mutation intervals. Those types of predictions could be aided by AI/ML models, which would further bolster the MTD mechanism capabilities in detecting adversarial action. Beware, changing these parameters is potentially disruptive. Additional research is required to ensure that MITM-alike attacks are not performed to steal the correct OTP for that timeslot without tricking the compensation mechanism into believing adversarial action is a regular network delay that must be accounted for. The mechanism does not suffer from this risk/threat in its present form because the baseline is fixed and determined in a controlled environment. However, an AI/ML system that unlocks further use-cases would need to be mindful of this issue to improve the protection already given by the MTD mechanism instead of inadvertently degrading it.

The incident response mechanism could also be used to eliminate false positives, by adding extra layers of validation in honeypot functions. Further research work is required to detect if user interactions with any given function are adversarial or legitimate, allowing for the seamless return from the honeypot to the real function (if and only if) the user is proven to be legitimate. Much like the previously suggested AI/ML compensation system, there is a fine line that must be trailed to achieve an improvement without inadvertently degrading the protection already given by the detection and response mechanism.

The orchestration side still has plenty of open opportunities. For starters, introducing an MTD mechanism will have computational requirements and potential adverse effects over other fundamental KPIs of the slice (*e.g.,* latency) that must be resolved before delivering a given security KPI value (*e.g., DynSec*). We still need further research into the policy definition that describes these rules. We also need policy arbitration work in scenarios with inter-slice communications. Network slices may also exist across administrative domains (interdomain or federation), requiring further study of these challenges (*e.g.,* different policy definition languages, other security requirements for compliance, and required capabilities baselines).

Using active monitoring and other monitoring data gathered by the Edge computing platform (as in 5G vertical systems) could significantly reduce the threat assessment system's false-negative rate. We could also define mobility constraints when used with mobile terminals and tune the MTD parameters accordingly. Crucially, integrating with power-constrained mobile terminals would likely require different resynchronization/recovery methods for the MTD mechanism. It would be crucial to assess the power efficiency of the mutations.

Lastly, we could pursue standardization of the TOTP-MTD approach and formally included it within a standardized framework. Such a work would require significant effort and the development of some improvements in this future work section so that the approach would work in more scenarios.

References

[1] S. Zhao, "The Internet and the Transformation of the Reality of Everyday Life: Toward a New Analytic Stance in Sociology," *Sociological Inquiry*, vol. 76, no. 4, pp. 458–474, Nov. 2006, ISSN: 0038-0245. DOI: 10.1111/j.1475-682X.2006.00166.x.

[2] K. Kasemsap, "The Importance of Electronic Commerce in Modern Business," in *Encyclopedia of Information Science and Technology, Fourth Edition*, IGI Global, pp. 2791–2801. DOI: 10.4018/978-1-5225-2255-3.ch243.

[3] M. K. Afzal, Y. B. Zikria, S. Mumtaz, A. Rayes, A. Al-Dulaimi, and M. Guizani, "Unlocking 5G Spectrum Potential for Intelligent IoT: Opportunities, Challenges, and Solutions," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 92–93, Oct. 2018, ISSN: 0163-6804. DOI: 10.1109/MCOM.2018.8493125.

[4] P. M. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2011. DOI: 10.6028/NIST.SP.800-145.

[5] M. Armbrust, I. Stoica, M. Zaharia, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, p. 50, Apr. 2010, ISSN: 00010782. DOI: 10.1145/1721654.1721672.

[6] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud Computing: An Overview," in *IEEE International Conference on Cloud Computing (CloudCom 2009)*, 2009, pp. 626–631. DOI: 10.1007/978-3-642-10665-1_63.

[7] ETSI ISG NFV, "Network Functions Virtualisation (NFV); Architectural Framework," *ETSI GS NFV 002 v1.2.1*, 2014.

[8] P. Demestichas, A. Georgakopoulos, D. Karvounas, *et al.*, "5G on the Horizon: Key Challenges for the Radio-Access Network," *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 47–53, Sep. 2013, ISSN: 1556-6072. DOI: 10.1109/MVT.2013.2269187.

[9] M. R. Palattella, M. Dohler, A. Grieco, *et al.*, "Internet of Things in the 5G Era: Enablers, Architecture, and Business Models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, Mar. 2016, ISSN: 0733-8716. DOI: 10.1109/JSAC.2016.2525418.

[10] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, Mar. 2017, ISSN: 1932-4529. DOI: 10.1109/MIE.2017.2649104.

[11] N. Kshetri, "The simple economics of cybercrimes," *IEEE Security & Privacy Magazine*, vol. 4, no. 1, pp. 33–39, Jan. 2006, ISSN: 1540-7993. DOI: 10.1109/MSP.2006.27.

[12] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The Characteristics of Cloud Computing," in *2010 39th International Conference on Parallel Processing Workshops*, IEEE, Sep. 2010, pp. 275–279, ISBN: 978-1-4244-7918-4. DOI: 10.1109/ICPPW.2010.45.

[13] I. Farris, T. Taleb, Y. Khettab, and J. S. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Communications Surveys Tutorials*, pp. 1–26, Aug. 2018, ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2862350.

[14] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2016.11.031.

[15] ONF, "SDN Architecture 1.1," *TR-521*, 2016.

[16] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, New York, New York, USA: ACM Press, 2013, p. 55, ISBN: 9781450321785. DOI: 10.1145/2491185.2491199.

[17] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving Target Defense Techniques: A Survey," *Security and Communication Networks*, pp. 1–25, Jul. 2018, ISSN: 1939-0114. DOI: 10.1155/2018/3759626.

[18] X. Zhou, Y. Lu, Y. Wang, and X. Yan, "Overview on Moving Target Network Defense," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, IEEE, Jun. 2018, pp. 821–827, ISBN: 978-1-5386-4991-6. DOI: 10.1109/ICIVC.2018.8492800.

[19] B. C. Ward, S. R. Gomez, R. Skowyra, *et al.*, "Survey of Cyber Moving Targets Second Edition," MIT Lincoln Laboratory Lexington United States, Tech. Rep., 2018.

[20] A. Aydeger, N. Saputro, and K. Akkaya, "A moving target defense and network forensics framework for ISP networks using SDN and NFV," *Future Generation Computer Systems*, vol. 94, pp. 496–509, May 2019, ISSN: 0167739X. DOI: 10.1016/j.future.2018.11.045.

[21] Y.-B. Luo, B.-S. Wang, X.-F. Wang, and B.-F. Zhang, "A keyed-hashing based self-synchronization mechanism for port address hopping communication," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 5, pp. 719–728, May 2017, ISSN: 2095-9184. DOI: 10.1631/FITEE.1601548.

[22] W. Connell, D. A. Menascé, and M. Albanese, "Performance Modeling of Moving Target Defenses," in *Proceedings of the 2017 Workshop on Moving Target Defense - MTD '17*, New York, New York, USA: ACM Press, 2017, pp. 53–63, ISBN: 9781450351768. DOI: 10.1145/3140549.3140550.

[23] W. Connell, D. A. Menasce, and M. Albanese, "Performance Modeling of Moving Target Defenses with Reconfiguration Limits," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018, ISSN: 1545-5971. DOI: 10.1109/TDSC.2018.2882825.

[24] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," *Proceedings of the 6th International Conference on Information Warfare and Security*, pp. 113–125, 2011.

[25] J. Andress, *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*, 2nd. Syngress Publishing, 2014, ch. 1, pp. 6–8, ISBN: 9780128008126.

[26] ISO/IEC JTC 1/SC 27, *ISO/IEC 27000:2018 - Information technology - Security techniques - Information security management systems - Overview and vocabulary*, 2018.

[27] R. Seebruck, "A typology of hackers: Classifying cyber malfeasance using a weighted arc circumplex model," *Digital Investigation*, vol. 14, pp. 36–45, Sep. 2015, ISSN: 17422876. DOI: 10.1016/j.diin.2015.07.002.

[28] ISO/IEC JTC 1/SC 27, *ISO 13335-1:2004 - Information technology – Security techniques – Management of information and communications technology security – Part 1: Concepts and models for information and communications technology security management*, 2004.

[29] P. Pols and P. Burghouwt, "The Unified Kill Chain," M.Sc. Thesis, Leiden University, 2017. [Online]. Available: https://www.csacademy.nl/en/csa-theses/february-2018/104-the-unified-kill-chain.

[30] B. D. Bryant and H. Saiedian, "A novel kill-chain framework for remote security log analysis with SIEM software," *Computers & Security*, vol. 67, pp. 198–210, Jun. 2017, ISSN: 01674048. DOI: 10.1016/j.cose.2017.03.003.

[31] S. Malone, "Using an expanded cyber kill chain model to increase attack resiliency," *Black Hat US*, 2016.

[32] ISO/TC 262, *ISO 31000:2018 - Risk management - Principles and guidelines*, 2018.

[33] P. K. Manadhata and J. M. Wing, "An Attack Surface Metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, May 2011, ISSN: 0098-5589. DOI: 10.1109/TSE.2010.60.

[34] P. K. Manadhata and J. M. Wing, "A Formal Model for a System's Attack Surface," in 2011, pp. 1–28. DOI: 10.1007/978-1-4614-0977-9_1.

[35] ISO/IEC JTC 1/SC 27, *ISO 27001:2013 - Information technology - Security techniques - Information security management systems - Requirements*, 2013.

[36] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., *Moving Target Defense* (Advances in Information Security). New York, NY: Springer New York, 2011, vol. 54, ISBN: 978-1-4614-0976-2. DOI: 10.1007/978-1-4614-0977-9.

[37] NIRTD, *National Cyber Leap Year Summit 2009 Participants' Ideas Report*, 2009. [Online]. Available: https://www.nitrd.gov/nitrdgroups/images/b/b8/Moving_Target_Summit_Ideas_Draft_090824_v3.pdf.

[38] NIRTD, *Cybersecurity Game-Change Research & Development Recommendations*, 2010. [Online]. Available: https://www.nitrd.gov/pubs/CSIA_IWG_%20Cybersecurity_%20GameChange_RD_%20Recommendations_20100513.pdf.

[39] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *National symposium on moving target research*, vol. 246, 2012.

[40] C. Lei, D.-H. Ma, H.-Q. Zhang, and L.-M. Wang, "Moving target network defense effectiveness evaluation based on change-point detection," *Mathematical Problems in Engineering*, 2016.

[41] ONF, "OpenFlow Switch Specification 1.0.0," *TS-001*, 2009.

[42] ONF, "OpenFlow Switch Specification 1.4.0," *TS-012*, 2013.

[43] ONF, "OpenFlow Switch Specification 1.5.0," *TS-020*, 2014.

[44] ONF, "Principles and Practices for Securing Software-Defined Networks," *TR-511*, 2015.

[45] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2453114.

[46] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE Way to Test OpenFlow Applications," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 127–140.

[47] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model checking invariant security properties in OpenFlow," in *2013 IEEE International Conference on Communications (ICC)*, IEEE, Jun. 2013, pp. 1974–1979, ISBN: 978-1-4673-3122-7. DOI: 10.1109/ICC.2013.6654813.

[48] ETSI ISG NFV, "Network Functions Virtualisation (NFV); Management and Orchestration," *ETSI GS NFV-MAN 001 V1.1.1*, 2014.

[49] S. W. Brim and B. E. Carpenter, *Middleboxes: Taxonomy and Issues*, RFC 3234, Feb. 2002. DOI: 10.17487/RFC3234. [Online]. Available: https://rfc-editor.org/rfc/rfc3234.txt.

[50] D. Naylor, K. Schomp, M. Varvello, *et al.*, "Multi-Context TLS (mcTLS)," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, New York, USA: ACM Press, 2015, pp. 199–212, ISBN: 9781450335423. DOI: 10.1145/2785956.2787482.

[51] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, "And Then There Were More: Secure Communication for More Than Two Parties," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies - CoNEXT '17*, New York, USA: ACM Press, 2017, pp. 88–100, ISBN: 9781450354226. DOI: 10.1145/3143361.3143383.

[52] ETSI ISG NFV, "Network Functions Virtualisation (NFV); NFV Security; Problem Statement," *ETSI GS NFV-SEC-001*, 2014.

[53] GSM Alliance, "An Introduction to Network Slicing," *White paper*, 2017.

[54] NGMN Alliance, "5G Security Recommendations Package #2: Network Slicing," 2016.

[55] 3GPP, "TR 33.811- Study on security aspects of 5G network slicing management," Tech. Rep. Release 15, 2018.

[56] Z. Kotulski, T. W. Nowak, M. Sepczuk, *et al.*, "Towards Constructive Approach to End-to-End Slice Isolation in 5G Networks," *EURASIP Journal on Information Security*, no. 2, pp. 1–23, 2018, ISSN: 2510-523X.

[57] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, vol. 1, IEEE Comput. Soc, pp. 176–185, ISBN: 0-7695-1212-7. DOI: 10.1109/DISCEX.2001.932214.

[58] R. L. Rivest, "The RC5 encryption algorithm," in *International Workshop on Fast Software Encryption*, Springer, 1994, pp. 86–96.

[59] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation," in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, New York, New York, USA: ACM Press, 2012, p. 127, ISBN: 9781450314770. DOI: 10.1145/2342441.2342467.

[60] S. Antonatos and K. G. Anagnostakis, "TAO: Protecting Against Hitlist Worms Using Transparent Address Obfuscation," in 2006, pp. 12–21. DOI: 10.1007/11909033_2.

[61] Y.-B. Luo, B.-S. Wang, X.-F. Wang, X.-F. Hu, and G.-L. Cai, "TPAH: a universal and multi-platform deployable port and address hopping mechanism," in *2015 International Conference on Information and Communications Technologies (ICT 2015)*, Institution of Engineering and Technology, 2015, ISBN: 978-1-84919-994-0. DOI: 10.1049/cp.2015.0230.

[62] Y.-B. Luo, B.-S. Wang, X.-F. Wang, B.-F. Zhang, and W. Hu, "RPAH: A Moving Target Network Defense Mechanism Naturally Resists Reconnaissances and Attacks," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 3, pp. 496–510, 2017, ISSN: 0916-8532. DOI: 10.1587/transinf.2016EDP7304.

[63] L. Jiang, Z. Hongqi, Y. Yingjie, and W. Yigong, "A Proactive Network Defense Method Based on Address Hopping for C/S Model," *Journal of Electronics & Information Technology*, vol. 39, no. 4, pp. 1007–1011, 2017.

[64] J. Sun and K. Sun, "DESIR: Decoy-enhanced seamless IP randomization," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, IEEE, Apr. 2016, pp. 1–9, ISBN: 978-1-4673-9953-1. DOI: 10.1109/INFOCOM.2016.7524602.

[65] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient Random Route Mutation considering flow and network constraints," in *2013 IEEE Conference on Communications and Network Security (CNS)*, IEEE, Oct. 2013, pp. 260–268, ISBN: 978-1-4799-0895-0. DOI: 10.1109/CNS.2013.6682715.

[66] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Formal Approach for Route Agility against Persistent Attackers," in 2013, pp. 237–254. DOI: 10.1007/978-3-642-40203-6_14.

[67] U. Rauf, F. Gillani, E. Al-Shaer, M. Halappanavar, S. Chatterjee, and C. Oehmen, "Formal Approach for Resilient Reachability based on End-System Route Agility," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense - MTD'16*, New York, New York, USA: ACM Press, 2016, pp. 117–127, ISBN: 9781450345705. DOI: 10.1145/2995272.2995275.

[68] S. Dolev and S. T. David, "SDN-Based Private Interconnection," in *2014 IEEE 13th International Symposium on Network Computing and Applications*, IEEE, Aug. 2014, pp. 129–136, ISBN: 978-1-4799-5393-6. DOI: 10.1109/NCA.2014.26.

[69] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for Moving Target Defense network protection," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, IEEE, Jun. 2014, pp. 1–6, ISBN: 978-1-4799-4786-7. DOI: 10.1109/WoWMoM.2014.6918979.

[70] Z. Zhao, F. Liu, and D. Gong, "An SDN-Based Fingerprint Hopping Method to Prevent Fingerprinting Attacks," *Security and Communication Networks*, pp. 1–12, 2017, ISSN: 1939-0114. DOI: 10.1155/2017/1560594.

[71] E. Al-Shaer, "Toward Network Configuration Randomization for Moving Target Defense," in 2011, pp. 153–159. DOI: 10.1007/978-1-4614-0977-9_9.

[72] M. Albanese, E. Battista, S. Jajodia, and V. Casola, "Manipulating the attacker's view of a system's attack surface," in *2014 IEEE Conference on Communications and Network Security*, IEEE, Oct. 2014, pp. 472–480, ISBN: 978-1-4799-5890-0. DOI: 10.1109/CNS.2014.6997517.

[73] M. Albanese, E. Battista, and S. Jajodia, "Deceiving Attackers by Creating a Virtual Attack Surface," in *Cyber Deception*, Cham: Springer International Publishing, 2016, pp. 167–199. DOI: 10.1007/978-3-319-32699-3_8.

[74] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Cyber Deception: Virtual Networks to Defend Insider Reconnaissance," in *Proceedings of the 2016 International Workshop on Managing Insider Security Threats - MIST '16*, New York, New York, USA: ACM Press, 2016, pp. 57–68, ISBN: 9781450345712. DOI: 10.1145/2995959.2995962.

[75] Y. Shi, H. Zhang, J. Wang, *et al.*, "CHAOS: An SDN-Based Moving Target Defense System," *Security and Communication Networks*, pp. 1–11, 2017, ISSN: 1939-0114. DOI: 10.1155/2017/3659167.

[76] Z. Zhao, D. Gong, B. Lu, F. Liu, and C. Zhang, "SDN-Based Double Hopping Communication against Sniffer Attack," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–13, 2016, ISSN: 1024-123X. DOI: 10.1155/2016/8927169.

[77] W. Connell, M. Albanese, and S. Venkatesan, "A Framework for Moving Target Defense Quantification," in *IFIP International Conference on ICT Systems Security and Privacy Protection*, Springer, 2017, pp. 124–138. DOI: 10.1007/978-3-319-58469-0_9.

[78] W. Connell, D. A. Menascé, and M. Albanese, "Performance Modeling of Moving Target Defenses," in *Proceedings of the 2017 Workshop on Moving Target Defense - MTD '17*, New York, New York, USA: ACM Press, 2017, pp. 53–63, ISBN: 9781450351768. DOI: 10.1145/3140549.3140550.

[79] X. Liang and Y. Xiao, "Game Theory for Network Security," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 472–486, 2013, ISSN: 1553-877X. DOI: 10.1109/SURV.2012.062612.00056.

[80] B. Van Leeuwen, W. M. Stout, and V. Urias, "Operational cost of deploying Moving Target Defenses defensive work factors," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, IEEE, Oct. 2015, pp. 966–971, ISBN: 978-1-5090-0073-9. DOI: 10.1109/MILCOM.2015.7357570.

[81] W. Connell, L. H. Pham, and S. Philip, "Analysis of Concurrent Moving Target Defenses," in *Proceedings of the 5th ACM Workshop on Moving Target Defense - MTD '18*, New York, New York, USA: ACM Press, 2018, pp. 21–30, ISBN: 9781450360036. DOI: 10.1145/3268966.3268972.

[82] G. Zhao, X. Xiong, and H. Wu, "A Model for Analyzing the Effectiveness of Moving Target Defense," in *Proceedings of the 8th International Conference on Communication and Network Security - ICCNS 2018*, New York, New York, USA: ACM Press, 2018, pp. 17–21, ISBN: 9781450365673. DOI: 10.1145/3290480.3290496.

[83] S. Lakshminarayana and D. K. Yau, "Cost-Benefit Analysis of Moving-Target Defense in Power Grids," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, Jun. 2018, pp. 139–150, ISBN: 978-1-5386-5596-2. DOI: 10.1109/DSN.2018.00026.

[84] M. Villarreal-Vasquez, B. Bhargava, P. Angin, *et al.*, "An MTD-Based Self-Adaptive Resilience Approach for Cloud Systems," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, IEEE, Jun. 2017, pp. 723–726, ISBN: 978-1-5386-1993-3. DOI: 10.1109/CLOUD.2017.101.

[85] S.-J. Moon, V. Sekar, and M. K. Reiter, "Nomad: Mitigating Arbitrary Cloud Side Channels via Provider-Assisted Migration," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, New York, New York, USA: ACM Press, 2015, pp. 1595–1606, ISBN: 9781450338325. DOI: 10.1145/2810103.2813706.

[86] M. Thompson, N. Evans, and V. Kisekka, "Multiple OS rotational environment an implemented Moving Target Defense," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, IEEE, Aug. 2014, pp. 1–6, ISBN: 978-1-4799-4187-2. DOI: 10.1109/ISRCS.2014.6900086.

[87] M. Thompson, M. Mendolla, M. Muggler, and M. Ike, "Dynamic Application Rotation Environment for Moving Target Defense," in *2016 Resilience Week (RWS)*, IEEE, Aug. 2016, pp. 17–26, ISBN: 978-1-5090-2002-7. DOI: 10.1109/RWEEK.2016.7573301.

[88] *PaX address space layout randomization (ASLR)*, https://pax.grsecurity.net/docs/aslr.txt, Accessed: 2019-05-31.

[89] J. Ganz and S. Peisert, "ASLR: How Robust Is the Randomness?" In *2017 IEEE Cybersecurity Development (SecDev)*, IEEE, Sep. 2017, pp. 34–41, ISBN: 978-1-5386-3467-7. DOI: 10.1109/SecDev.2017.19.

[90] A. Bittau, A. Belay, A. Mashtizadeh, D. Mazieres, and D. Boneh, "Hacking Blind," in *2014 IEEE Symposium on Security and Privacy*, IEEE, May 2014, pp. 227–242, ISBN: 978-1-4799-4686-0. DOI: 10.1109/SP.2014.22.

[91] S. W. Boyd, G. S. Kc, M. E. Locasto, A. D. Keromytis, and V. Prevelakis, "On the General Applicability of Instruction-Set Randomization," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 3, pp. 255–270, Jul. 2010, ISSN: 1545-5971. DOI: 10.1109/TDSC.2008.58.

[92] K. Sinha, V. P. Kemerlis, and S. Sethumadhavan, "Reviving instruction set randomization," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, May 2017, pp. 21–28, ISBN: 978-1-5386-3929-0. DOI: 10.1109/HST.2017.7951732.

[93] X. Jiang, H. J. Wangz, D. Xu, and Y.-M. Wang, "RandSys: Thwarting Code Injection Attacks with System Service Interface Randomization," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, IEEE, Oct. 2007, pp. 209–218, ISBN: 0-7695-2995-X. DOI: 10.1109/SRDS.2007.36.

[94] S. H. Kim, L. Xu, Z. Liu, Z. Lin, W. W. Ro, and W. Shi, "Enhancing Software Dependability and Security with Hardware Supported Instruction Address Space Randomization," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, IEEE, Jun. 2015, pp. 251–262, ISBN: 978-1-4799-8629-3. DOI: 10.1109/DSN.2015.48.

[95] Chao Zhang, Tao Wei, Zhaofeng Chen, *et al.*, "Practical Control Flow Integrity and Randomization for Binary Executables," in *2013 IEEE Symposium on Security and Privacy*, IEEE, May 2013, pp. 559–573, ISBN: 978-0-7695-4977-4. DOI: 10.1109/SP.2013.44.

[96] K. Koning, H. Bos, and C. Giuffrida, "Secure and Efficient Multi-Variant Execution Using Hardware-Assisted Process Virtualization," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, Jun. 2016, pp. 431–442, ISBN: 978-1-4673-8891-7. DOI: 10.1109/DSN.2016.46.

[97] P. Ammann and J. Knight, "Data diversity: an approach to software fault tolerance," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 418–425, Apr. 1988, ISSN: 00189340. DOI: 10.1109/12.2185.

[98] M. Christodorescu, M. Fredrikson, S. Jha, and J. Giffin, "End-to-End Software Diversification of Internet Services," in 2011, pp. 117–130. DOI: 10.1007/978-1-4614-0977-9_7.

[99] W. Jiangxing, "Research on Cyber Mimic Defense," *Journal of Cyber Security*, vol. 4, pp. 1–10, 2016. DOI: 10.19363/j.cnki.cn10-1380/tn.2016.04.001.

[100] H. Hu, J. Wu, Z. Wang, and G. Cheng, "Mimic defense: a designed-in cybersecurity defense framework," *IET Information Security*, vol. 12, no. 3, pp. 226–237, May 2018, ISSN: 1751-8709. DOI: 10.1049/iet-ifs.2017.0086.

[101] H. Hu, Z. Wang, G. Cheng, and J. Wu, "MNOS: a mimic network operating system for software defined networks," *IET Information Security*, vol. 11, no. 6, pp. 345–355, Nov. 2017, ISSN: 1751-8709. DOI: 10.1049/iet-ifs.2017.0085.

[102] B. Ma and Z. Zhang, "Security research of redundancy in mimic defense system," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, IEEE, Dec. 2017, pp. 2910–2914, ISBN: 978-1-5090-6352-9. DOI: 10.1109/CompComm.2017.8323064.

[103] W. Liu, F. Chen, H. Hu, G. Cheng, S. Huo, and H. Liang, "A Novel Framework for Zero-Day Attacks Detection and Response with Cyberspace Mimic Defense Architecture," in *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, Oct. 2017, pp. 50–53, ISBN: 978-1-5386-2209-4. DOI: 10.1109/CyberC.2017.39.

[104] ENISA, "Threat Landscape for 5G Networks: Updated Threat assessment for the fifth generation of mobile networks (5G)," Tech. Rep. December, 2020. DOI: 10.2824/802229.

[105] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600935.

[106] R. F. Olimid and G. Nencioni, "5G Network Slicing: A Security Overview," *IEEE Access*, vol. 8, pp. 99 999–100 009, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2997702.

[107] N. Alomar, P. Wijesekera, E. Qiu, and S. Egelman, ""You've Got Your Nice List of Bugs, Now What?" Vulnerability Discovery and Management Processes in the Wild," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, USENIX Association, Aug. 2020, pp. 319–339, ISBN: 978-1-939133-16-8.

[108] T. Petsas, G. Tsirantonakis, E. Athanasopoulos, and S. Ioannidis, "Two-factor authentication: is the world ready?" In *Proceedings of the Eighth European Workshop on System Security - EuroSec '15*, New York, New York, USA: ACM Press, 2015, pp. 1–7, ISBN: 9781450334792. DOI: 10.1145/2751323.2751327.

[109] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," RFC Editor, RFC 4226, Dec. 2005.

[110] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," RFC Editor, RFC 6238, May 2011. DOI: 10.17487/rfc6238.

[111] ETSI ISG NFV, "Network Functions Virtualisation (NFV); Management and Orchestration," *ETSI GS NFV-MAN 001 V1.1.1*, 2014.

[112] Y.-B. Luo, B.-S. Wang, X.-F. Wang, B.-F. Zhang, and W. Hu, "RPAH: A Moving Target Network Defense Mechanism Naturally Resists Reconnaissances and Attacks," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 3, pp. 496–510, 2017, ISSN: 0916-8532.

[113] L. Szekeres, M. Payer, Tao Wei, and D. Song, "SoK: Eternal War in Memory," in *2013 IEEE Symposium on Security and Privacy*, IEEE, May 2013, pp. 48–62, ISBN: 978-0-7695-4977-4. DOI: 10.1109/SP.2013.13.

[114] *CVE-2017-0144*. Available from MITRE, CVE-ID CVE-2017-0144. Mar. 2017. [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2017-0144.

[115] V. A. Cunha, D. Corujo, J. P. Barraca, and R. L. Aguiar, "Moving Target Defense to set Network Slicing Security as a KPI," *Internet Technology Letters*, May 2020, ISSN: 2476-1508. DOI: 10.1002/itl2.190.

[116] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: Simpler, Smaller, Fast as MD5," in *Applied Cryptography and Network Security*, Springer, Berlin, Heidelberg, 2013, pp. 119–135. DOI: 10.1007/978-3-642-38980-1_8.

[117] H.-G. Berns, T. Burnett, R. Gran, and R. Wilkes, "GPS time synchronization in school-network cosmic ray detectors," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 848–853, Jun. 2004, ISSN: 0018-9499. DOI: 10.1109/TNS.2004.829368.

[118] D. Mills, J. Martin, J. Burbank, and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC 5905 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5905.txt.

[119] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, 2020. DOI: 10.1109/IEEESTD.2020.9120376.

[120] A. Doğanaksoy and F. Göloğlu, "On Lempel-Ziv Complexity of Sequences," in 2006, pp. 180–189. DOI: 10.1007/11863854_15.

[121] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. Alcaraz Calero, "Virtual IoT HoneyNets to Mitigate Cyberattacks in SDN/NFV-Enabled IoT Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1262–1277, Jun. 2020, ISSN: 0733-8716. DOI: 10.1109/JSAC.2020.2986621.

[122] P. Quinn, U. Elzur, and C. Pignataro, *Network Service Header (NSH)*, RFC 8300, Jan. 2018. DOI: 10.17487/RFC8300. [Online]. Available: https://rfc-editor.org/rfc/rfc8300.txt.

[123] A. Binder, T. Boros, and I. Kotuliak, "A SDN Based Method of TCP Connection Handover," in *Information and Communication Technology - EurAsia Conference (ICT-EurAsia)*, 2015, pp. 13–19. DOI: 10.1007/978-3-319-24315-3_2.

[124] W. Fan and D. Fernandez, "A novel SDN based stealthy TCP connection handover mechanism for hybrid honeypot systems," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–9.

[125] W. Fan, Z. Du, M. Smith-Creasey, and D. Fernandez, "HoneyDOC: An Efficient Honeypot Architecture Enabling All-Round Design," *IEEE Journal on Selected Areas in Communications*, 2019, ISSN: 0733-8716.

[126] S. Kyung, W. Han, N. Tiwari, *et al.*, "HoneyProxy: Design and implementation of next-generation honeynet via SDN," *IEEE Conference on Communications and Network Security (CNS)*, Oct. 2017. DOI: 10.1109/CNS.2017.8228653.

[127] W. Han, Z. Zhao, A. Doupé, and G.-J. Ahn, "HoneyMix: Toward SDN-based Intelligent Honeynet," *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization - SDN-NFV Security*, 2016. DOI: 10.1145/2876019.2876022.

[128] R. Berthier and M. Cukier, "Honeybrid: A hybrid honeypot architecture," in *USENIX Security Symposium*, vol. 2008, 2008.

[129] S. L. Pfleeger and R. K. Cunningham, "Why Measuring Security Is Hard," *IEEE Security & Privacy Magazine*, vol. 8, no. 4, pp. 46–54, Jul. 2010, ISSN: 1540-7993.

[130] S. Xu, "Cybersecurity Dynamics: A Foundation for the Science of Cybersecurity," *Proactive and Dynamic Network Defense, Springer*, 2019.

[131] H. Chen, J.-H. Cho, and S. Xu, "Quantifying the security effectiveness of firewalls and DMZs," *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security - HoTSoS*, 2018.

[132] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," RFC Editor, RFC 2104, Feb. 1997.

[133] 3GPP, *Management and orchestration; Provisioning, TS 28.531, V15.4.0*, Sep. 2019.

[134] 3GPP, *Management and orchestration; 5G Network Resource Model (NRM), TS 28.541, V16.3.0*, Dec. 2019.

[135] V. A. Cunha, N. Maroulis, C. Papagianni, *et al.*, "5growth: Secure and reliable network slicing for verticals," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit): Network Softwarisation (NET) (2021 EuCNC & 6G Summit - NET)*, Porto, Portugal, Jun. 2021.

[136] V. A. Cunha, D. Corujo, J. P. Barraca, and R. L. Aguiar, "TOTP Moving Target Defense for sensitive network services," *Pervasive and Mobile Computing*, vol. 74, p. 101 412, 2021, ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2021.101412.

[137] H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, RFC 2104 (Informational), Updated by RFC 6151, Internet Engineering Task Force, Feb. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2104.txt.