Detection and Mitigation of DDoS/Dos Security threats in an NFV Architecture

Jorge Steven Martinez osorio

Tesis de maestría presentada para optar al título de Magíster en Ingeniería de Telecomunicacionesr

Director
Prof. Dr.-Ing. Juan Felipe Botero Vega

Universidad de Antioquia
Facultad de Ingeniería
Maestría en Ingeniería de Telecomunicaciones
Medellín, Antioquia, Colombia
2022

Cita	Martinez Osorio Jorge Steven, 2022 [1]
Referencia Estilo IEEE (2020)	[1] Martínez Osorio J. S. “Detection and Mitigation of DDoS/Dos Security threats in an NFV Architecture”, Tesis de maestría, Maestría en Ingenieria de Telecomunicaciones, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.

Maestría en Ingeniería de Telecomunicaciones, Cohorte XIV.
Grupo de Investigación Telecomunicaciones Aplicadas (GITA).

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Biblioteca Carlos Gaviria Díaz

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.
Decano/Director: Jesús Francisco Vargas Bonilla.
Jefe departamento: Augusto Enrique Salazar Jiménez.

Acknowledgments

This project would not have been possible without the support of many people. I would also like to extend my deepest gratitude to my adviser, Juan Felipe Botero, who read my numerous revisions and helped make some sense of the confusion and offered guidance and support to make it possible. I also wish to thank to Jaime Alberto Vergara, who was an important partner during this process offering your knowledge and valuable advice to improve this work.

Thanks to the University of Antioquia for awarding me a Master Fellowship, providing me with the a huge part financial means to complete this project. And finally, thanks to my girlfriend, parents, and numerous friends who endured this long process with me, always offering support and love.

Abbreviations

BIC Bayesian Information Criterion.

DDoS Distributed Denial of Service.

DoS Denial of Service.

DPI Deep packet inspection.

EM Expectation Maximization Algorithm.

ETSI European Telecomunications Standards institute.

GMM Gaussian Mixture Model.

IDS Intrusion Detection System.

LB Load Balancer.

MANO Management and Orchestration.

ML Machine Learning.

NFV Network Functions Virtualizations.

NFVI Network Functions Virtualization Infrastructure.

NS Network Service.

ONF Open Network Foundation.

OSM Open Source Mano.

OVS Open Virtual Switch.

PDF Probability Density Function.

RF Random Forest.

SDN Software Defined Networking.

SLA Service Level Agreement.

SP Service Provider.

UBM Universal Backgfround Model.

VIM Virtual Infrastructure Manager.

VM Virtual Machine.

VNF Virtual Network Function.

VNFM Virtual Network Function Manager.

List of Figures

List of Tables

Contents

Abstract

DDoS/DoS attacks are one of the most used attacks by cybercriminals. Due to their huge impact in traditional or novel network architectures, these kinds of attacks can make that the biggest websites fail.

The novel Network Functions Virtualizations (NFV) architecture can also be affected by the external attacks, and the DDoS/DoS also affect the NFV layers, being the Network Functions Virtualization Infrastructure (NFVI) the most critical layer as it hosts the major part of the attack that also affect the other layers. This fact makes the NFV architecture an interesting target for the attackers.

There are many different kinds of traditional techniques used for DDoS/DoS attack detection, some of them include Artificial Intelligence, Intrusion Detection Systems (IDSs), Deep packet inspection (DPI). Most of them are well known and have remained unchanged during the last few years. In this work, we implement a novel technique called Gaussian Mixture Model (GMM), normally used in other scientific or engineering areas, to detect DDoS/DoS cyberattacks in a real NFV environment.

Moreover, this work developed a mitigation strategy to avoid the negative impact caused by DDoS/DoS attacks, inside the Software Defined Networking (SDN)-NFV environment.

Finally, this work presents an additional strategy as a complement to the aforementioned mitigation strategy to cover all aspects that can affect Web service availability. This strategy looks for the implementation of a load balancer to distribute the network traffic through a pool of servers to avoid the situation in which thousands or millions of users sent requests to the Web service and provoking, denial of service with legitimate traffic.

As a results, this work proves that the novel Machine Learning (ML) technique (GMM) implemented to prevent the attack was very powerful blocking around 1.3 million of DDoS/DoS packets (this amount of traffic represents around 90% of the incoming traffic in this test) sent by the attacker, allowing the Web server to continue to provide the service without any interruption. Also, the load balancing strategy was able to cover and manage situations with a huge volume of traffic requests sent to a Web server and proving that it is capable to preserve the service availability and the benefit of using it was over 36% much efficient in contrast to not using it.

This work performs the implementation of the previously mentioned strategies and shows their benefits in a real NFV environment where the system was able to mitigate the DDoS/DoS attacks and avoid the negative impact caused by thousands of users, guaranteeing the service availability exposed in the NFV environment.

1 Introduction

For almost three decades, the Internet has been accepted and used widely; nowadays it supports abundant technologies and distributed applications [4]. To cope with the growing complexity demanded by the modern Internet applications, Service Providers (SPs) have implemented many different technologies causing a slow evolution of networks, as all stakeholders must find an agreement to improve or change the architectures and adopt new technologies [5]. To implement and deploy these new technologies, offering new services and maintaining service level agreements, SPs have increased the Operational and Capital expenditures (Opex and Capex) due to the need of installing new specialized hardware (also known as middleboxes) [6].

For the last few years, an emergent paradigm from the industry is changing the way SP deploy, manage and operate network services; it is called NFV [2]. With NFV, the middleboxes from traditional networks are managed as simple modules of software aiming to decouple the network functions from the proprietary hardware in order to use general purpose commodity hardware [6, 2]. This approach allows the SP to create, configure and maintain network services in an easier way. It also provides an easy way to deploy new network services in few days at a lower cost reducing the Capex. The reduction of Opex is also possible thanks to the implementation of more efficient operation, maintenance and updating of the network functions. These actions can be performed remotely and at scale, allowing to improve the efficiency, agility, resource utilization and scalability of the network services offered by SP (see Fig. 1) [2, 7].

In November 2012, the European Telecomunications Standards institute (ETSI) was selected to be the home of the Industry Specification Group for NFV (ETSI ISG NFV). Since this date, this community is responsible for updating the following items: architectural NFV framework, computation descriptions, hypervisor, NFV infrastructure, and Management and Orchestration (MANO), among other concepts [8, 9, 1].

Taking into account the growth of information technology, several threads and vulnerabilities have been used by the attackers to get some benefits. It is possible due to the fact that cyberattacks are cheaper and the attackers use the anonymous nature of the Internet to perform their attacks. Cybersecurity addresses multiple issues including different kind of technologies creating countermeasures to mitigate or reduce the impact of those attacks aiming to preserve the confidentiality, integrity, and availability of the information [10].

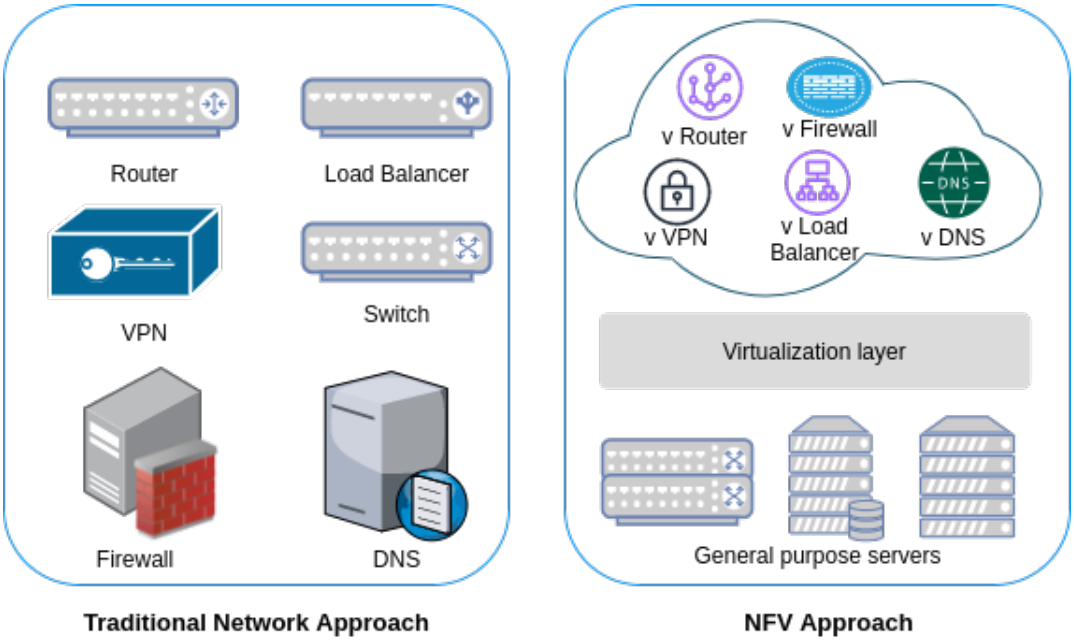


Fig. 1. NFV approach vs traditional approach.

In spite of the effort to cover the major part of threads on traditional technologies, cybersecurity concerns have gained attention, since some threats and vulnerabilities are still not tackled in NFV MANO. For instance, there is no definition of a standard interface in the ETSI NFV architecture to deploy virtual security functions reacting to various threats in real time [2, 11]. Also, handling DDoS or DoS [12] and the triggering of automatically orchestrated security functions in the NFVI are still ongoing research topics [2].

The previous open issues are some of the many challenges that must be addressed to provide an NFV platform with a set of tools to detect and mitigate security threats and implement countermeasures for any type of security attack in NFV environments. One of the most important or vulnerable sectors is the NFVI. NFVI is where the major part of security threats and vulnerabilities in the NFV environment are found [13, 2]. This shows the NFVI as a critical point to implement strategies to detect and mitigate attacks. Any effort to develop countermeasures in NFV is a significant support to consider in order to improve the security in these new technological environments.

Taking into account this background, our main aim in this research work is to develop a strategy to detect and mitigate an attack to the NFVI and prevent that other Virtual Network Function (VNF) could be attacked.

To carry out this proposal, we selected a cyberattack called DDoS/DoS, that is very common but also represents and important risk in terms of information availability. To address this attack in a NFV environment, it is necessary to define a technique to detect it. The proposed techniques are a Gaussian Mixture Model classifier and Universal Background Model which are widely used in speech recognition applications as generic probabilistic models for speaker detection or verification [14, 15]. We proposed this GMM classifier to explore how new classification mechanisms, used for other applications, perform in the detection of DDoS attacks [16] and it will be used to detect the DDoS/DoS attack and normal traffic inside a NFV environment. Also, we explore the Universal Backgfround Model (UBM) classifier because this is a large GMM containing a relatively big set of population features which could show interesting results.

The GMM and UBM classifiers are interesting because these strategies in general terms try to model the extracted features of the phenomenon as a mixture of Gaussians, offering a different way of analyzing the data. Attack and benign traffic could be modeled by a set of Gaussians and each new flow in the network should match one of the two models to determine whether it is malicious or normal traffic. It is important to mention that both GMM and UBM have different ways of determining whether incoming traffic in a time window transformed to a mixture of Gaussians is an attack or not.

To evaluate the GMM and UBM classifiers we will make a comparison with a traditional Random Forest (RF) classifier, allowing us to have a preliminary remarks about the experiments. After making this traffic classification, we define an strategy to mitigate DDoS/DoS attacks and prevent future troubles with the same thread.

The mitigation strategy consists of installing flow rules on an SDN switch to drop those malicious traffic during a established time.

In summary, our contributions are focused on:

- This work implements from scratch a private cloud computing platform using Open Source Mano (OSM) and OpenStack and defines an starting point for every future work inside the GITA research group that needs a private cloud or a real NFV environment to test or develop new research ideas over a complete NFV environment.
- Also, this work develops a strategy to detect one of the most common cyberattacks on the industry (DDoS/DoS) through a novel ML technique in networking (called

GMM) to perform a wise attack detection.

- This work develops a mitigation strategy deploying a set of drop flow rules to avoid the failure of the compromised VNF inside the Network Service (NS).
- Finally, this work shows an easier way to add other strategies to improve the performance of a VNF and a NS such as implementing a load balancer that is enabled just when the strategy detects a high throughput and this high throughput does not match as DDoS/DoS attack. This additional strategy works as an evaluation scenario for the detection strategy.

2 Background

This section shows a set of concepts such as NFV, SDN, MANO, VNF, network security, and the selected ML technique called GMM aiming to subsequently present the main contributions of this work.

2.1 Network Functions Virtualization (NFV)

Network Functions Virtualization represents the implementation of data plane network functions (traditionally implemented in specialized hardware), in software able to run in commodity hardware; the main idea of NFV is the decoupling of the network functions from the physical (specific-purpose) devices [17, 7]. This paradigm offers new possibilities to manage the network infrastructure resources more efficiently and gives SPs more flexibility in network capabilities, allowing to deploy or support new services faster and cheaper with more agility.

The decoupling of both parts (network functions and physical infrastructure) helps to reassign and share physical resources, allowing several functions to run at the same time in the same commodity host or data center. This decoupling allows the innovation and evolution of these two parts independently and improves their lifecycle [7].

NFV deployment will reduce the Capex and Opex in contrast to traditional infrastructure, where network functions are provided by specific-vendor hardware [17]. This allows that the telecommunications market becomes more dynamic with new operative focuses, as the network functions are managed as simple virtual modules.

On the other hand, traditional network infrastructure is often forced to integrate devices with specific purposes, implying higher complexity in network operation and maintenance. In addition, for traditional network infrastructure, hardware's lifecycle ends quickly, implying that the cycle of acquisition, integration, and deployment is repeated continuously [18, 6]. the comparison between NFV and a traditional architecture is shown in Fig. 2.

Some advantages of NFV are:

- Reduction of energy consumption.
- Less time for developing and offering new network services.

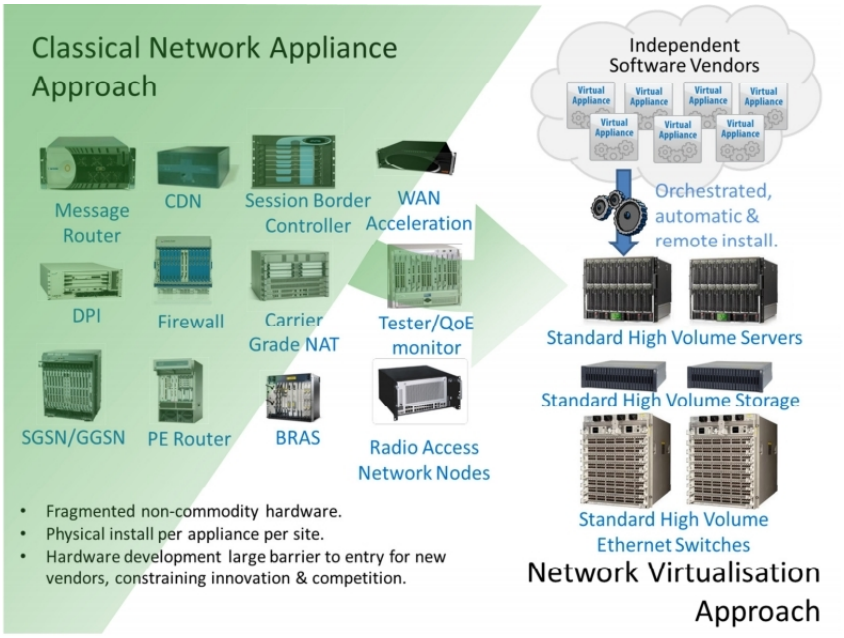


Fig. 2. NFV architecture vs traditional architecture [1].

- Deployment of new services with reduced risk.
- Flexibility and dynamism would permit the development of demand-oriented services.
- Reduction of Capex and Opex.

NFV aims to transform the perspective of the network operators using the evolution of virtualization technology, in order to consolidate different network functions in general purpose hardware [18]. Fig. 3 shows the NFV architecture proposed by ETSI that is composed by the NFVI, VNF, and application or business layers.

2.2 Virtual Network Function (VNF)

To explain a VNF, we first define a Network Function (NF) as a functional block inside the network infrastructure that has external interfaces and a well defined functional conduct. NFs are known in traditional networks as middleboxes. The VNFs are NFs that run on virtual resources from commodity hardware like Virtual Machines (VMs) [7]. An example of some NFs can be firewalls, IDS, load balancers, among others [17].

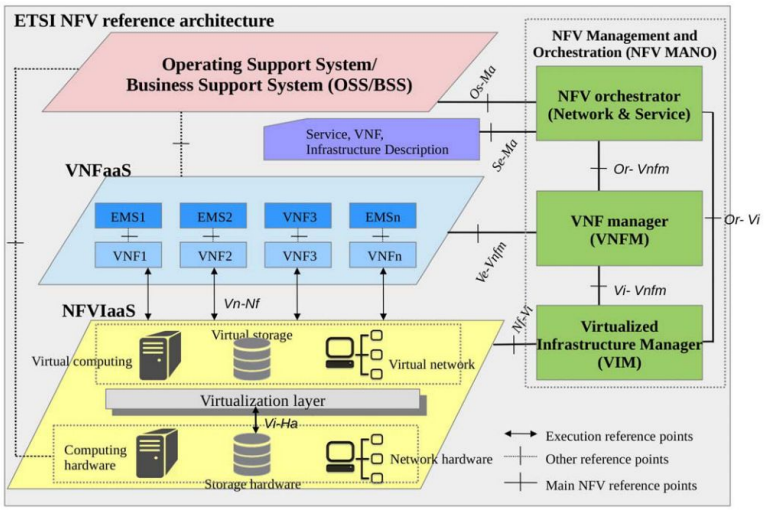


Fig. 3. NFV architecture MANO proposed by ETSI NFV framework, taken from [2].

2.3 Management and orchestration (MANO)

With the implementation of NFV, a new set of MANO functions became necessary due to decouple NFs from physical devices. These new MANO functions allow to create new dependencies between NFs and physical devices to ease the network management and to provide a more scalable and flexible architecture. To achieve that, a set of features are required:

- Interoperable standard interfaces.
- Common information models.
- Mapping of information models to data models.

MANO is used to provide the functionality required to configure the VNFs, to manage the physical and/or logical resources, and to support the infrastructure virtualization and NFVs lifecycle. Furthermore, MANO includes the databases used to store the information and data models that define the deployment, lifecycle properties, services and resources [6].

As Fig. 3 shows, NFV-MANO is composed of the orchestrator, the Virtual Network Function Manager (VNFM)), and the Virtual Infrastructure Manager (VIM) [7].

- The orchestrator is responsible for managing and supporting the databases, way points, and interfaces used to exchange information among all components. This information can be used by multiple managers to apply changes in NFVI or in VNFs.
- A VNFM has control over single or multiple VNF instances, and manages their

lifecycle.

- The VIM is used to control and manage the physical resources and machines that are part of NFVI under a single authority domain. There could exist more than one VIM in a NFV architecture in order to manage a specialized NFVI.

2.4 Load Balancer (LB)

A load balancer is a network function used to distribute the network traffic through a set of servers that could handle multiple request within the same service. A common load balancer follows a set of steps to handle the incoming request:

- It receives multiple requests from different kinds of clients.
- It builds the request queue to manage the incoming requests.
- It monitors the load status of the servers on the set of available servers.
- It uses the chosen load balancing strategy/algorithm/heuristic to send the incoming request to the selected server.

A load balancer technique helps to increase the resource availability for an specific service, reduce the amount of request that could be handled by a unique server and makes it possible to have a huge range of flexibility when the service needs to increase its availability [19].

The most common load balancing strategies are:

- **Least-Connection Load Balancing**: When a new request arrives to the Load Balancer (LB), the LB chooses the server with least active connections.
- **Weighted Least-Connection Load Balancing**: It is an extension of least-connection load balancing where each server is given by a wight among 0-1, and the server with 1 will not receive more requests.
- **Dynamic Feedback Load Balancing Scheduling**: The LB takes into account information of availability, load information and response time. these information are extracted periodically to send incoming request to the server pool.
- **Round-Robin Load Balancing**: This load balancer gives to each server in the server pool one request to handle. The server number one in the pool servers will receive the next request to handle when the last server inside the pool receives its request.
- **Weighted Round-Robin Load Balancing**: This LB is an extension of the round robin LB where servers are given different weight numbers according to their

capability. Each server will receive request first if its weight is more higher than other.

2.5 Software Defined Networking (SDN)

In traditional networks the design, configuration and network management have high complexity and depend on manual intervention in each device (using low-level commands) [20]. These current networks are integrated with a vertical model of control and data plane, that are bundled inside the network devices [21]. The high complexity and manual intervention can result in possible failures, making the networks hard to manage and upgrade, which implies that flexibility, scalability, and evolution are not possible [20].

SDN is born in the first half of the 1990s with active networks [22, 3], and some prominent works are RCP [23], 4D [24] and Ethane [20], where the goal was to give the capability to the network to perform actions that can be programmed.

SDN is an initiative that arises as an emergent network paradigm, which can give possible solutions to the traditional networks problems. SDN breaks the vertical integration of control plane (takes the decisions of how to manage the network traffic) and data plane (forwards the traffic and takes the actions established by the control plane), decoupling both and establishing a logically centralized control of the network (controller or network operating system-NOS). This simplifies policy implementation and network reconfiguration, allowing the innovation and evolution of networks, and tries to solve the current problems in traditional networks, making it easier to create and to add new network abstractions [21, 3]. The SDN architecture is depicted in Fig. 4.

Two concepts have been introduced to deploy SDN in the industry. These concepts are the southbound and northbound interfaces. The first one represents the interface and protocols that perform the communication between programmable switches and the software controller. The second one is like a middleware between the application layer and the controller that determines the operational tasks, network policies and translation of these policies for the controller to interpret them correctly [25, 3].

The most important southbound interface at the moment is OpenFlow [26], which is considered as the defacto standard defined by the Open Network Foundation (ONF). Many vendors have included it in their programmable switches. There are many OpenFlow controllers such as NOX [27], POX [28], ONIX [29], Floodlight [30], OpenDayLight [31], Ryu [32] and Onos [33]. Which are developed in different programming

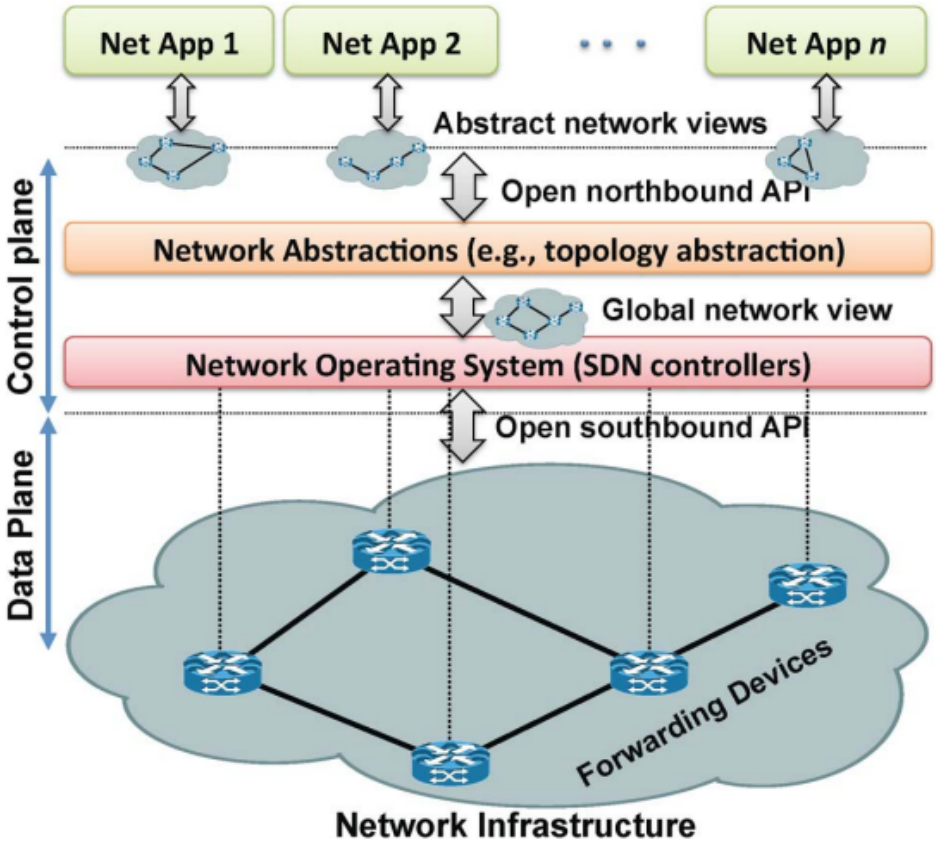


Fig. 4. SDN architecture, taken from [3].

languages: C++, Java or Python. Selecting one of these depends on the robustness and response time needed in the application.

2.6 SDN and NFV

NFV is an emerging technology, which can be highly complementary with SDN, but is important to clarify that they do not depend on each other, and both can be implemented separately; however, their combination can enhance their performance[18].

With the current techniques used in data centers, NFV can be accomplished, but with the separation of control and data plane by SDN; NFV can facilitate the operations and maintenance procedures and simplify the compatibility with ongoing deployments. Additionally, NFV and SDN are closely aligned to use commodity hardware [18, 6, 8].

SDN can play an important role in NFV infrastructure and orchestration of resources where some features as provisioning and configuration of network connectivity, bandwidth provisioning, security, and policy control are relevant. This join effort can speed up deployment improving the flexibility and automation. With SDN, we can program logical functions that can be adapted to the NFVI [1, 7].

2.7 Security on data networks

A security attack on communication networks may refer to any effort to expose, disable, disturb, destroy or access private information without authorization [34]. Security can be defined as a set of guidelines to apply on the network design and their future expansion, aiming to minimize the leak, disturb or improper use of the information that the communication networks manage.

The principal concepts on security data networks are:

- Availability: It represents the response time that must be guaranteed for any device inside the network; this device must be ready to perform its functions without the additional processing. To make this possible is necessary to raise countermeasures and mechanisms to prevent failures in different devices in the network without adverse consequences in the performance, and accessibility in case of any emergency.

- Integrity: It guarantees that the information life cycle (generation, transmission, and receives) remains genuine even if any attacker (internal or external) tries to change or modify it. This process can be defined as authenticity and non-repudiation, thus, is necessary to discard false information and to implement countermeasures.

- Confidentiality: It refers to the fact that no external agent can have access to the network information without the necessary credentials. Mechanisms must exist to perform the validation of the information in real-time and continuously in order to guarantee that sensible information does not leak at any time affecting the normal behavior of system.

To get more details about these concepts, please refer to [35].

2.8 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model is a probabilistic model aiming to depict data as of a Gaussian distribution or a combination of multiple Gaussian distributions [36]. This technique is widely used in speech recognition where it is important to detect a specific speaker or validate a set of speakers inside a given application.

A Gaussian distribution is composed by a random variable X following a Gaussian Probability Density Function (PDF) in one dimension:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{x-\mu^2}{2\sigma^2}) \doteq N(x;\mu,\sigma^2) \tag{1}$$

Where μ is the mean and σ is the standard deviation of $\boldsymbol{X}$. This distribution is widely used in different areas as science or engineering and can be used to generalize problems with different nature [37]. The multivariate case of the Gaussian distribution follows the next equation:

$$f(\boldsymbol{x}) = \frac{1}{\boldsymbol{\Sigma}^{\frac{1}{2}}\,(2\pi)^{\frac{D}{2}}} exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})) \doteq N(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) \tag{2}$$

Where $\boldsymbol{\mu}\,\epsilon\,\mathbb{R}^D$ is the means vector and it is composed of the mean of each dimension of the random variable $\boldsymbol{X}$. Also, $\boldsymbol{\Sigma}\,\epsilon\,\mathbb{R}^{D\times D}$ is the covariance matrix and represents the variance of the random variable in a higher dimension.

The GMM can be used to represent data as a linear combination of a number of Gaussian distributions, where, for example, each distribution could represent a subpopulation of the dataset, and by joining all these distributions,a good representation of the whole population can be obtained. The GMM can be generalized from Eq. 2:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \frac{c_m}{|\boldsymbol{\Sigma_m}|^{\frac{1}{2}}\,(2\pi)^{\frac{D}{2}}} exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_m})^T{\boldsymbol{\Sigma_m}}^{-1}(\mathbf{x}-\boldsymbol{\mu_m})) \tag{3}$$

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m\, N(\boldsymbol{x};\boldsymbol{\mu_m},\boldsymbol{\Sigma_m}) \tag{4}$$

$$\sum_{m=1}^{M} c_m = 1 \tag{5}$$

Eq. 3 shows that a GMM is the sum of M Gaussian distributions (or multivariate Gaussian distributions), each one of them weighted by the parameter c_m. This parameter must be less or equal than 1 and the sum over all c_m must be equal to 1. The number of Gaussian distributions can be determined in different ways. One alternative is running a grid search, where different GMM are compared according to accuracy results obtained in the training and test steps. On the other side, an Information criterion like the Bayesian Information Criterion (BIC) or the Akaike Information Criterion can be used to understand how well the model represents the data [38].

In order to estimate the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, and c_m of the distribution, the GMM algorithm uses the Expectation Maximization Algorithm (EM). This algorithm iterates between two steps to calculate and refine the parameters that increase the model's likelihood. In the first step, called the expectation step or E step, the parameters to determine can be randomly initialized or a clustering algorithm can be used, after this probability of a sample belonging to a particular Gaussian m in the iteration j is calculated. Then, in the second step called the maximization step or M step, a new set of parameters $\boldsymbol{\Lambda}(\boldsymbol{\mu_m}, \boldsymbol{\Sigma_m}, \boldsymbol{c_m})$ that maximizes the aforementioned probability is chosen. These two steps will be repeated until the probability difference between two iterations is less than a certain threshold [39, 40].

2.9 Universal Background Model (UBM)

In a general form, the UBM is a large GMM containing a relatively big set of population features. There are different ways to obtain the final model. One of them, and probably the easiest one, is to generate a large GMM using all the data of the training step. The classes must be balanced to avoid biases over the dominant class. Another way to obtain the final model is to train an individual UBM for each class in the training dataset and then mix all of these subpopulation models. The latter approach has the possibility of using unbalanced data and controlling the composition of the final model [14].

After the UBM is generated, an adaptation method must be used to fit the trained model to the new data and to generate the final model, taking a part of the training data and the trained model. This is done by executing the Maximum *A Posteriori* (MAP) estimation. MAP will be applied over the UBM to estimate a class dependent model [41]. MAP provides an alternative to maximum likelihood estimation for machine learning. Also, MAP tries to estimate the updated distribution and the parameters $\boldsymbol{\Lambda}$ that explain in a better way the population maximizing the *a posteriori* probability without the need of calculating it. According to the Bayes theorem the *a posteriori* probability can be calculated by:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{6}$$

Commonly this equation can be referred to as the *a posteriori* probability of A given B and the $P(A)$ is the *a priori* probability of A. $P(B)$ is used to normalize the result, it can be removed and the *a posteriori* now is proportional to the probability of $P(B|A)$ multiplied by $P(A)$.

$$P(A|B) \propto P(B|A)\ P(A) \tag{7}$$

MAP estimation is not interested in calculating the exact value or the *a posteriori* probability, instead, MAP is interested in optimize $P(B|A)\ P(A)$, aiming to estimate the distribution and the parameters $\mathbf{\Lambda}$ which could explain in the best way the phenomenon.

$$maximize\ P(\mathbf{\Lambda}|\boldsymbol{x}) = P(\boldsymbol{x}|\mathbf{\Lambda})\ P(\mathbf{\Lambda}) \tag{8}$$

The GMM and UBM techniques will be used to perform the experimental analysis as they provide a different way of modeling the network data based on a mixture of Gaussians where we do not know anything about the distribution of the data nor what the behavior of the data is. Furthermore, these techniques could be used to perform a classification analysis to detect benign or malicious traffic, or even they could even be used to perform an anomaly detection where we model only a well known traffic (legitimate traffic) and in the case of detecting any anomaly we could take some action with the traffic such as isolating it, blocking it, among others.

2.10 Random Forest

Random Forest (RF) is a machine learning algorithm composed of predictive models called decision trees which are formed by binary rules where is possible to allocate the samples according to their attributes and predict the value of the response variable. Each decision tree is trained with a slightly different sample in comparison with the other decision trees. These samples with slight differences are obtained by a bootstrapping method that samples the training dataset and assign these samples to the decision trees defined by the random forest model [42, 43].

Random forest obtains a new prediction taking into account the predictions of all the decision trees. RF is capable of choosing the features from the dataset to use in the model automatically. RF is used in regression and classification problems and it

does not require a lot of data pre-processing. On the other hand, RF can not be used to extrapolate results with different features in the training step. Also, the model's interpretability can be lost using multiple decision trees. RF will be used as a baseline to compare the performance of GMM and UBM in the attack classification task.

3 State of the Art

As a small context, the principal goal of the NFV environment is to provide the user with a set of tools that helps him/her to: build an underlying infrastructure, build secure services, have an optimal management and orchestration of his/her architecture, have a flexible infrastructure, and make an easier maintainability of all NFV layers. This means, to provide a novel way to design, build, manage and orchestrate services starting from the definition of the required infrastructure until release the network service according to the particular needs.

For SPs that have implemented a NFV environment, these tasks represent several challenges and complexity to ensure the requirements that must be satisfied for each tenant. One of the most important objectives is to provide well-defined security policies, mechanisms, and countermeasures in order to guarantee the Service Level Agreements (SLAs) for each tenant [2, 13].

The following subsections are going to present a set of threats and vulnerabilities that have been detected by different authors in the literature showing a set of different proposals or countermeasures to avoid such threats.

3.1 Principal threats and vulnerabilities on NFV environments

According to the authors in [2], the NFV environment has several challenges, open issues, threats, and vulnerabilities inevitably induced due to the fact that the traditional attack vectors could take action in NFV and also, new threats and vulnerabilities could appear when the attackers take advantage of this new architecture. To cover these concerns, the administrator of the NFV platform must implement countermeasures to detect and mitigate many different actions performed by the attackers when they try to modify, break the platform, leak the data, gain access, execute malicious code or trigger any other suspicious action.

Several works have tried to solve or present novel strategies to guarantee the SLAs, trustworthiness of the platform, and security levels of an NFV environment [2, 13, 12, 11, 44, 45, 46, 47]. For instance, the authors in [2] present an exhaustive work in security threat analysis, best practices, and countermeasures to prevent and mitigate the cyberattacks and improve the security in the NFV architecture. They highlight that the possible vulnerabilities in the NFV environment can be even larger than the ones of traditional architecture. Some vulnerabilities in NFV can be due to: orchestration of policies, administrative errors, deployment of corrupted VNFs on new

hardware, domain administration, and frequent migration. These vulnerabilities open the landscape to possible attack vectors and several challenges in the identification and enforcement of security policies. These issues are related to the fast development of NFV, which makes it difficult to implement a robust defense security system.

In [2], the authors analyze different scenarios with possible threats and vulnerabilities. Then, they show different use cases and what kind of threats or vulnerabilities can appear, allowing the SP to deploy cost-effective security according to his/her needs.

To present the detected threats and vulnerabilities on the NFV architecture is necesary to define the NFV layers showed in Fig. 3:

- NFVI layer: It concentrates the issues related to the hypervisor, VM and hardware vulnerabilities.
- VNF layer: It concentrates the issues related to network service vulnerabilities.
- NFV MANO layer: It concentrates the issues related to the management, orchestration and control plane attacks, isolation failures, policy violation, and lack of interoperability.

We have organized the approaches [2, 13, 12, 11, 44, 45, 46, 47] according to the NFV layer as follows:

- **NFVI layer**: Authors in [2] talk about **NFV infrastructure as a service (NFVIaaS)** where the NFVI provides storage, network processing capabilities, and fundamental computing resources. It allows the customers to run arbitrary network services without managing the underlying infrastructure. Guest VM, can be exploited with malicious code and can be a victim of DDoS or Man in the Middle attacks (among many others). Also, an attacker can:
  - Violate the isolation of a VM.
  - Manipulate memory and execute operations in another VM.
  - Gain access to the hypervisor getting root privileges.
  - Gain access to a VM.
  - Break the hypervisor by sending packets to induce buffer overflow.
  - Gain root privileges to run malicious code.

  If the hypervisor is compromised, a new hypervisor could run on top of the original one, putting at risk different VMs. Bad configurations of the hypervisor could

lead to giving more resources to an attacker and enabling the attacker to assault other guests VM.

Authors in [12] show several security challenges in the NFV environment, which includes problems in the NFVI such as:

- Network elasticity.
- Unauthorized access.
- Data leakage on hypervisor domain.
- Security threats in shared computing resources (CPU, Memory).
- Privilege control.
- Network configurations.
- Security policies.

Different threats and vulnerabilities such as insecure management interfaces, compromising virtual network components, security pitfalls of OpenStack, inadequate enforcement of security policies, shared physical and virtual resources could appear.

The work in [11] explains the pros and cons of the NFV architecture implementation. They argue that possible flaws in the underlying infrastructure can affect the system's resiliency and the quality of the offered services. The principal focus must be the virtual infrastructure manager, where the hypervisor is the main element to protect.

Some potential risks are:

- Isolation failure risk: Here the attacker tries to gain access to the hypervisor through an infected VNF that runs over the hypervisor, affecting the NFV architecture and its functionality.
- Network topology validation and implementation failures: Avoiding the utilization of validation tools could lead to errors in highly dynamic services.
- Security logs troubleshooting failures: These kinds of attacks can overload the log files on the hypervisor, making the logs hard to analyze for any VNF that runs over the hypervisor. Also, if the logs are leaked, an attacker could extract sensitive data.

Some of the possible and most critical attacks that could be deployed in this layer are the DDoS/DoS, Man in the Middle, sniff attacks, spoofing attacks and malicious Troyan code. These attacks are caused by different reasons like bad design or configuration that allows the attacker to use more resources than allowed, insecure management interfaces, inadequate enforcement of security policies, improper isolation between hypervisor and VNFs, among others.

- **NFV MANO**: Authors in [2, 12] talk about **Virtual Network Platform as a Service (VNPaaS)** that allows an enterprise to develop and deploy their custom services to meet their business purpose. Some of the threat vectors for VNPaaS could be exploiting default application configuration, flaws of SSL/TLS, security breaches resulting from lack of interoperability, security flaws in the development life cycle, and Malicious insiders.

  Some of the possible and most critical attacks that could be deployed in this layer are DDoS/DoS, backdoor attacks, attacking public key infrastructure (PKI). These attacks are caused by different reasons like exploiting default application configuration, flaws of SSL/TLS, flaws in development life cycle, flaws in control plane and management, among others.

- **VNF**: Authors in [2] talk about **Virtual Network Function as a Service (VNFaaS)**: Here, the enterprise is a customer that can configure the applications without having control over the underlying infrastructure; it is similar to VNPaaS, but the difference is the scale of service and programmability. Here we can find significant threats because the SP does not expose the internal infrastructure and users do not have control over network resources.

  Some of the possible and most critical attacks that could be deployed in this layer are DDoS/DoS, bypass firewall restrictions and spoofing attacks. These attacks are caused by different reasons like flaws in VNF software, lack of interoperability, flaws in security policies, improper isolation between hypervisor and VNFs, among others.

Summarizing, the major part of the NFVI attack vectors are replicated over the other NFV layers and the DDoS/DoS attack appears in all layers.

3.2 Proposals and countermeasures

To solve the mentioned problems, the authors in [12] also mention some security solutions existing in the industry to provide and enforce security in NFV:

- Policy Manager for NFV [48].
- Virtualized security at the network edge: A user-centric approach [49].
- VMware vCloud NF [50].

These platforms propose different solutions trying to mitigate or solve problems like enforcing security policies, providing safe and efficient allocation of data flows across VNFs, and processing and analyzing historical and real-time data.

Furthermore, the authors in [11] also present a set of best practices to implement in an NFV architecture in order to prevent possible threats and vulnerabilities. Here, they mention some mechanisms to enforce security and mitigate possible attacks such as the implementation of a trusted platform module to verify and store the measurements of sensitive components of the system, validating the platform measurements before performing any action. Also, they propose to disable all the unused services to prevent possible threats and vulnerabilities, creating isolated traffic zones or traffic for similar functionalities to protect these data using access control policies. Finally, they propose to protect the data of VNF volumes/swap areas using encryption and storing the cryptographic keys in safe locations.

In [2],the following mechanisms are shown to compare the different implementations between traditional network scenarios and NFV environments in the aforementioned aspects.

- Security Management and orchestration
- Identity and Access Management (IAM)
- Intrusion Detection and Prevention (IDS/IPS)
- Network Isolation
- Data Protection

In each mechanism, the authors perform a comparison between a traditional implementation and a NFV based implementation, analyzing some aspects such as flexibility, centralized control and management, the complexity of lifecycle management, scalability, cost, effectiveness, and security enhancement. Some security countermeasures and recommendations in NFVI Layer, VNF Layer, and NFV MANO

Layer are presented to guarantee the security and protection of data in case of any attack. Also, they propose a framework structure trying to cover all security threats and vulnerabilities, using the most relevant countermeasures to improve security in a NFV environment, this framework is depicted in Fig. 5.

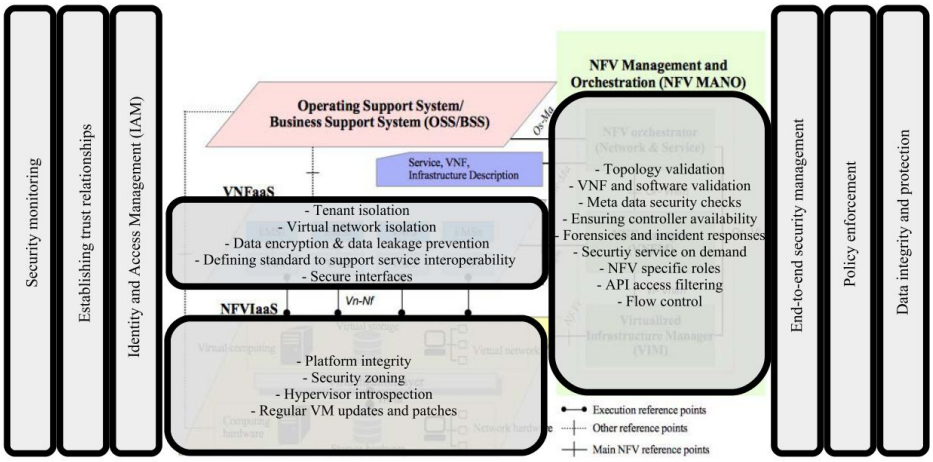


Fig. 5. NFV MANO framework structure proposed by [2]

Also, authors in [51] propose a ML technique to manage traffic and detect attacks on the NFV architecture. Their solution works as an agent which receives different network metrics and builds network profiles according to these metrics to perform different actions. In this case, the authors implement two profiles, one of them is a strategy used to detect and mitigate DoS attacks and the other profile is used to perform a load balancing strategy if the throughput of normal traffic in the environment increases. This work deploys a prototype using Docker to implement the network functions, POX as SDN controller, and Mininet to emulate network topologies. It is an interesting approach as this work deploys the detection and mitigation strategy to prevent possible troubles caused by the attack and generates an additional profile to guarantee the service availability with high throughput. This work inspired us to develop our proposal; however, it would have been very interesting to see this work in a real scenario to take it as a point of reference to compare it results with other proposals.

3.3 Classification and taxonomy

To summarize this chapter, we have proposed a taxonomy based on a clear classification criterion, this section shows a set of tables that aim to condense the reviewed literature in a clear way.

Table 1 exposes a review of security threats and vulnerabilities found in the literature. The review has been classified into three layers, NFVI, VNFs, and NFV MANO, and shows the possible attack vectors in each layer.

Table **??** collects the countermeasures and the mechanisms that can be implemented in an NFV environment to prevent and mitigate the security issues and vulnerabilities presented in these three layers. And finally, Table 3 presents the open issues and challenges still present in the NFV environment.

In synthesis, all authors have proposed a compilation of threats and vulnerabilities in three principal aspects. The most critical vulnerabilities are found in NFVI because they are shared in more than one layer, and affect the upper layers of a NFV environment. To prevent these problems, it is necessary to enforce the mechanisms and policies used in NFVI layer, to reduce the volume and impact of attack vectors.

Here, we note that there are many open challenges and issues to address and solve in the future. Probably, a long time is needed to adopt a standardized framework to cover the most important concerns in NFV security. All of these reviewed works show a huge challenge principally in NFVI, as depicted in Table 1, where the major part of the threats and vulnerabilities to address are allocated.

Table 1. Security threats and vulnerabilities in each layer (NFVI, VNF, NFV MANO).

NFV layer	Security threats and vulnerabilities	possible Attacks	Literature
NFVI	Data leakage. Violate the isolation between guest VM. Manipulation of memory allowing to write, read and execute in other VM. Break the hypervisor to gain access and run any code or put a new hypervisor on top. Bad design or configuration that allows the attacker to use more resources than allowed. Insecure management interfaces. Inadequate enforcement of security policies. Shared physical and virtual resources. Infected VM images. Compromising VM migration. Bad manage and inappropriate operations. Lack of security evaluation and integration and interoperability among different software. Improper isolation between hypervisor and VNFs. Create a quick and dynamic service decision with virtual network component are a mistake. Service disruption or unavailability by exhaust the hypervisors or physical resources. Leaked of infrastructure logs. Platform integrity. Data transmitted unencrypted. Malicious administrator. Unsuitable service composition (flaws security protection).	DDoS, malicious Troyan code, Man in the middle, sniff attack, Hyperjacking, insider attack, spoofing attack, DNS amplification attack, eavesdropping attack	[2, 13, 12, 11, 44, 45, 46, 47]
VNF	Flaws in VNF software. Lack of interoperability. Flaws in development life cycle. Flaws in security policies. Insecure interfaces. Data loss and information leakage. Malicious insiders. Flaws in management and control plane. Problems in fault detection and troubleshooting. Improper isolation between hypervisor and VNFs. Data transmitted unencrypted.	bypass firewall restrictions, DDoS, spoofing attack, VM escape attack, eavesdropping attack	[2, 13, 12, 11, 44]
NFV MANO	Exploiting default application configuration. Flaws of SSL/TLS. Breaches in lack of interoperability and inconsistency in management and orchestration (between multi-vendor software, diverse computational resources and sophisticated isolation by enterprises and operators). Flaws in development life cycle. Malicious insiders. Insecure interfaces. Exploit insecure VNF API to dump the records of personal data from the database to violate user privacy. Malicious administrator. Flaws in control plane and management. Lack of well-defined policies and isolation.	DDoS, backdoor attacks, attacking public key infrastructure (PKI), hyperjacking, VM escape, VM hoping	[2, 13, 12, 11, 44, 45, 47]

3.4 DDoS/DoS in NFV environments

For this work, we will select a DDoS/DoS attacks as it is a very common and critical security threat that is transversal to all NFV layers to find an efficient strategy to detect and mitigate this security threat on NFV environments and to explore a different way to address this kind of threats in NFV environments. The idea is to open the possibilities to implement and deploy NFV platforms for different future projects. This attack was selected as it affects the availability of NS and could even affect the entire NFV environment.

In Tables 1, **??**, and 3, we presented the principal security issues in an NFV environment. There we remarked the critical weakness through the NFV layers (NFV MANO, VNF, NFVI), also, Chapter 3 shows that the NFVI is the most critical layer where we could find the major part of threats and vulnerabilities. Additionally, a lot of these threats and vulnerabilities are transversal to the NFV layers, which means that if we cover the principal issues on the NFVI layer, we will cover a high part of cybersecurity troubles and will maintain a safe NFV environment.

For this reason, this work will be oriented to develop a tool that performs Detection and mitigation in an NFV environment to prevent malicious actions, maintain the information safe, and guarantee an adequate performance of the different architectures and services implemented within the platform.

For this purpose, our work will consist on the proposal of detection and mitigation techniques for DDoS/DoS attacks in NFV environments, remarking that handling DDoS/DoS attacks and the triggering of automatically orchestrated security functions in the NFVI are still ongoing research topics [2]. Our proposal is based on a work of the Canadian Institute for cybersecurity of the University of New Brunswick that performed a research using real network traffic to study different kind of attacks with different types of targets like Windows or Linux machines. They present a robust dataset created using a tool called CICFlowMeter [52, 53] developed by them. This tool models DDoS/DoS attacks with current malicious software, including other malicious attacks with other behaviors.

Therefore, our goal is to develop a tool to detect and mitigate DDoS/DoS attacks using the dataset released for the Canadian Institute for cybersecurity and implement a ML technique to detect a DDoS/DoS attack in a real NFV environment. Furthermore, this work will develop a strategy to maintain the performance of the network service deployed inside of the NFV environment where there is not a volumetric attack but the throughput through the NS gets over a certain boundary.

Table 2. Countermeasures and Mechanisms proposed to improve the security in the NFV environment.

	NFV Covered Layer		
Mechanisms and countermeasures proposed	**NFVI**	**VNF**	**NFV MANO**
Allows Advanced security service deployment to achieve high-level agility and efficient service deployment (mechanism), [2]		✓	✓
Improving capability of defending against massive attacks using metadata, limited bandwidth, etc. (mechanism), [2]	✓		✓
Automation and central management of security functions according to the common policies (mechanism), [2]			✓
Security orchestrator based on Moon framework to provide policy enforcement and validating security characteristics (mechanism), [2, 13]			✓
Has design principles as specify high level policies, achieving fine-grained security control and provide a set of security functions (mechanism), [2]		✓	✓
Virtualized access control based on federated identity to provide flexible management and increase security (mechanism), [2]			✓
Virtualization of content-aware identity and access control management to cover all necessary requirements including privileged user management, fine- grained access control, among others (mechanism), [2]			✓
Security monitoring appliance to provide active, transparent, and real time security monitoring (mechanism), [2]	✓	✓	
Encrypted virtual machine introspection to provide users a complete status of their virtual instances, while keeping confidentiality of user's data by using encryption technique (mechanism), [2]	✓	✓	
Security health monitoring – CloudMonatt to provides a flexible distributed cloud architecture to detect and monitor the security health of customer's VMs based (mechanism), [2]	✓	✓	
Virtualized network isolation to provide traffic isolation in a virtualized datacenter environment (mechanism), [2, 13]	✓		
Virtualized network isolation to provide traffic isolation in a virtualized datacenter environment (mechanism), [2]	✓		
Logical isolated network partitions to maintain end-to- end isolated network virtualization (mechanism), [2]	✓		✓
Data leakage detection – CloudSafetyNet is a propose to examine the data flow of tenant applications in the cloud platform whether there is any data leakage (mechanism), [2]		✓	✓
Data confidentiality protection, architecture that proposes to protect user privacy and ensure that service providers are not able to collect any user's data (mechanism), [2]	✓	✓	✓
Key-insulated symmetric key cryptography tries to mitigate the repeated exposure of secret keys, in which the keys need to be updated frequently (mechanism), [2]	✓		✓
Key management in the clouds – EnCloud tries to solve a set of problems of insecure key creation, key management, and key renewal (mechanism), [2]		✓	
Decomposing services for data plane and control plane, enforcing policies and virtualizing resources for control functions, and managing and controlling the whole network to improve the security in NFV (mechanism), [2]	✓		
Establishing trust domain to provide confinement boundaries, create secure communication, and maintain security among group's members (countermeasure), [2, 44]	✓		
Dynamic and adaptive access control to protect, monitoring and enforce the policies to access the data stored across NFVI (countermeasure), [2, 13]	✓		
Hypervisor introspection or Virtual Machine Introspection is used to scrutinize software running inside the VMs, and then analyze suspicious events and anomalous activities (countermeasure), [2, 11]	✓		
Separation of administrative duties to identify what administrative roles should be separated and what these can do (countermeasure), [2]	✓		
Security service chaining aims to maintain closed-loop security protection or to achieve autonomic controllability (countermeasure), [2, 44]	✓		
Regular VM updates and patches to reduce vulnerabilities and mitigate security risks from VM attacks (countermeasure), [2, 13, 11]	✓		
Remote attestation, with this technic/process the user is able to verify whether a queried cloud platform is booted in a trusted manner (mechanism/countermeasure), [2, 11, 46]	✓		

	NFV Covered Layer		
Mechanisms and countermeasures proposed	**NFVI**	**VNF**	**NFV MANO**
Design of trustworthy hardware and platform to provide a secure execution environment in the processor supply chain and the hardware mechanism, tries to improve security in virtualization, among others (countermeasure), [2]	✓		
Defining standards to support service interoperability to support interoperability between the various NFV elements, covering both standalone VNF instances, NFVI platforms, and network functions (countermeasure), [2, 13]		✓	
Ensuring robustness of service chaining to ensure availability of each independent network services in service chaining (countermeasure), [2]		✓	
Establishing trust relationships between intra, inter, and extra domains to ensure that all parties in the same trust domain can securely communicate with each other (countermeasure), [2, 13]		✓	
Security monitoring to provide visibility into the network events and activities of interest the status of infrastructure resources, network services, VNF instances, and other related events (countermeasure), [2, 13]	✓	✓	
Data encryption and leakage prevention to protect data in transit, at rest, and in use (countermeasure), [2, 12]		✓	
VNF image signing and software integrity protection, aims to verify that the images don't being infected with malwares or others attacks (countermeasure), [2, 11]		✓	
Security management, orchestration, and automation to ensure secure and fast service delivery, while improving end user experience (countermeasure), [2, 13, 11]			✓
Transparency to network control and management to ensure the confidentiality of data and it can only operate with the user permission, make the environment trusted (countermeasure), [2]			✓
Access control virtualized network function to verify the large number of VNF appliance (authentication and authorization) (/mechanism/countermeasure), [2, 12]		✓	✓
Have strict operational procedures for people to mitigate internal threats from inappropriate operations (countermeasure), [12]	✓		
The devices should have a security certification process to eliminate possible threats from design and implementation (countermeasure), [12]	✓		
NFVI should has and adopt standard security mechanisms for authentication, authorization, encryption, and validation (countermeasure), [12]	✓		
Secure networking techniques should be adopted, such as TLS, IPSec, and SSH to provide security in shared logical networking layer and shared physical network interface controllers (countermeasure), [12]	✓		
Defines standard interfaces (authentication, user privilege control, among others) to implements these in virtualized security functions (challenge), [12]	✓	✓	
Decomposing services for data plane and control plane, (2) enforcing policies and virtualizing resources for control functions, and (3) managing and controlling the whole network (countermeasure), [12]	✓	✓	✓
Trusted platform module (TPM) to verify and store measurement of system sensitive components and validates the platform measurements before performed (mechanism), [11, 46]	✓		
Create and isolate zone traffics or similar functionalities and protect these data with access control policies and dedicated firewall based (mechanism), [11, 13]	✓	✓	
Disables all services that are not in use to prevent possible threats and vulnerabilities (mechanisms), [11]	✓	✓	
Use Secure virtualization (sVirt) to integrate mandatory access control security with Linux based hypervisors, provides isolation between VM processes and data files (mechanism), [11]	✓		
Protect the data of VNF volumes/swap areas using encryption and storing the cryptographic keys in a safe locations (mechanism), [11]	✓	✓	
Define an up/down resource optimization to provide a higher level of scalability (mechanism/countermeasure), [44]	✓	✓	
Enable the resilience, survivability and ensure the availability of security services (mechanism), [44]	✓	✓	✓

	NFV Covered Layer		
Mechanisms and countermeasures proposed	**NFVI**	**VNF**	**NFV MANO**
protecting integrity of a virtualized infrastructure managed by a *Cloud Service Provider* (OpenCIT) allowing the following aspects: Integrity of the NFVI host platform, Integrity and confidentiality of VNFs, and Integration of trust with the MANO stack (mechanism), [45, 46]	✓		✓
Uses a external *trusted security orchestrator* to perform VNF related security operations (mechanism), [46]			✓
Uses a Security orchestrator as an extension of NFV orchestrator (Moon-based) (mechanism), [47, 13]			✓
implement a strategy to provide security service on demand (countermeasure), [13]	✓	✓	
Ensuring controller availability because is the centralized decision point of an NFV environment (countermeasure), [13]			✓

Table 3. Open issues and challenge In the NFV environment.

Open security challenge	**Proposed by**
Define the standard interface in the ETSI NFV architecture to deploy virtual security functions to react to various threats in real time	[11]
Perform the trust management between different vendors who build NFVs hardware and software	[11]
Run time attestation is still an open research area that needs to be explored further.	[11]
What is the level of intelligence that must be achieved in NFV MANO, more importantly, a basic set of security functions should be automatically orchestrated and intelligently deployed to the appropriate NFV Infrastructure for protecting network assets based on the predefined policies	[2]
Find the best process that ensures the correct configuration of network security technologies in intra and inter policies avoiding conflicts	[2]
The risks of unauthorized disclosure of their data performing by untrusted service providers could be high. The challenge is that users have to ensure the service providers do not break down their privacy and confidential data.	[2]
(Compromised NFV) How to detect compromised components and mitigate their impact remains a challenge	[12]
(Distributed Denial-of-service attacks) How to utilize the flexibility of VNF to defend against DDoS attacks in the network	[12]
(Trust management in NFV) how to manage the trust chain and evaluate the trustworthiness of products	[12]
How to adaptively configure VNFs by choosing software to minimize security risk of the network	[12]
The implementation of Trusted Computing methodologies by the Trust Manager should be assessed because of the extensive use of virtualization in NFV (attestation of VNFs is a relevant topic from the research perspective as it encompasses the challenges into establishing trust for different virtualization technologies)	[45]
Attestation of container-based VNFs is a promising area for further research because of the lack of production-ready solutions that apply to different OS virtualization technologies	[45]
VNF image integrity and confidentiality	[45]
Inside the Relationship between trust and the MANO stack, Trust Manager should be able of providing limited visibility of attestation reports to the other components of the MANO stack depending on the assignment of hosts to specific tenants.	[45]
Impact on performance and scalability of trust operations because is introduced a latency during the measurement and attestation phases, and the verification phase requires the remote verifier.	[45]
TelcoCloud has a requirement to remain resilient and fault tolerant under extreme circumstances to maintain service level agreements (SLAs) in the following scenarios: 1) Unavailability of resources that satisfy a policy 2) VNF integrity is compromised 3) Binding of VNF resulted in failure 4) NFV platforms are not trusted	[45]
trusting the infrastructure focusing on the requirements for trusted hardware and a trusted virtualization layer	[46]

4 DDoS/DoS attack detection and mitigation proposal in NFV/SDN Environments

This Chapter will show an introduction about the traditional ML techniques used to detect DDoS/DoS attacks, also, it will show the feature extraction process and data prepossessing, the description of each experimental scenario, and the performance evaluation obtained on each experiment. additionally, this chapter will show a preliminary conclusions after analyzing the experimental results and the proposed mitigation process to face the DDoS/DoS attacks or tackle the availability problems provoked by high volume of requests.

4.1 DDoS/DoS detection process

This subsection will show preliminary experiments and results to subsequently define the DDoS/DoS detection proposal.

4.1.1 Traditional ML-based detection techniques

The intrusion detection inside the computer networks is an important topic to address due to the risk suffered by the information. Focusing on the prevention and detection of DoS/DDoS attacks over computer networks. These works are using traditional IDS or traditional machine learning (ML) techniques. The following works show different approaches and methodologies to detect the aforementioned attacks :

The authors in [54] generate a huge IDS data-set called CICIDS2017, used in this work that contains a different kind of cybersecurity attacks, and then, the authors perform an analysis to determine the best feature sets and detect different attacks using common ML methods like Random Forest, KNN, Naive-Bayes, among others.

They present good results as a preliminary evaluation of the data-set presented in the paper. In [55] the authors show a huge amount of intrusion detection techniques from traditional IDSs and ML and neural network techniques. This work gives a good base to take into account to work in this area.

In [56] the authors present a technique based on a nonparametric cumulative sum (CUMSUM) oriented to detect DoS attack in web servers where the DoS attacks have different behaviors with a high volume of traffic or low volume of traffic in the application layer.

The authors in [57] proposed an IDS using a convolutional neural network technique to detect DoS attacks and compare the performance of it with other techniques as

KNN, SVM, and Naive-Bayes.

Additionally, In [58] the authors present a smart DoS/DDoS detection, designed to detect high and low Dos attacks. This work uses random traffic samples collected on the network devices using stream protocol and random forest technique.

All these works propose interesting ideas to address the detection of DoS/DDoS cybersecurity attacks but most of them are based on traditional ML techniques. For this reason, we proposed a different approach using a GMM and UBM technique as a novel method to detect DoS/DDoS cybersecurity attacks inside a network.

4.1.2 Data-set and Feature Extraction

This section explains how the used dataset was built starting from the captured data and how the feature extraction over these data was performed.

The dataset used in this work is **"Intrusion Detection Evaluation Dataset (CICIDS2017)"**, it contains benign network traffic and updated common security cyberattacks. The authors ensure that the dataset resembles the true real-world data, and CICFlowMeter was used to extract data from traffic flows [59]. They added 85 statistical features, but more can be included [60, 61]. Table 4 shows some of these features.

The dataset was built on the behavior of 25 users generating HTTP, HTTPS, FTP, SSH, and email traffic, these flows are labeled as BENIGN. The dataset also includes attacks like Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. This work uses only the data that contains DoS/DDoS and normal traffic included in a 12.5GB pcap file. The original dataset has 692703 samples with both traffic labels/classes and 85 features. These features were reduced to 68 deleting features that provide little information. The removed features helped to identify the traffic flow but do not provide information about the network phenomenon.

The dataset has 251712 samples with the label **"DDoS/DoS"** and 439683 samples with the label **"Benign"**. In order to use a balanced dataset to construct unbiased GMM/UBM, 251712 samples were randomly selected from the BENIGN traffic. Next, standardization and a correlation analysis were performed. Some of the features were correlated, so a dimension reduction via Principal Component Analysis (PCA) was conducted, ending with 18 components that contain 77.69% of the variance from the original dataset. The explained variance Analysis is depicted in Fig. 6 where the sum over the first 18 components gives the aforementioned variance.

Table 4. Features generated by the CICFlowMeter tool.

Feature Name	Description
Flow duration	Duration of the flow in Microsecond
Total Fwd Packet	Total packets in the forward direction
Total Bwd packets	Total packets in the backward direction
Total Length of Fwd Packet	Total size of packet in forward direction
Total Length of Bwd Packet	Total size of the packet in backward direction
Fwd Packet Length Min	Minimum size of the packet in forward direction
...	...
std idle	Standard deviation time a flow was idle before becoming active
Init Win bytes forward	The total number of bytes sent in initial window in the forward direction
Init Win bytes backward	The total number of bytes sent in initial window in the backward direction
Act data pkt forward	Count of packets with at least 1 byte of TCP data payload in the forward direction

4.1.3 Experimental scenarios

This section shows how was the dataset split in each experimental scenario and what technique was used in each experiment.

For the UBM approach, 50% of the dataset will be used to build the UBM. After this, 30% will be used for training, which includes the generation of GMMs for benign traffic and for attack traffic and the implementation of the MAP algorithm. Finally, the remaining 20% will be used for the testing phase. This is summarized in Table 5.

Table 5. Dataset segmentation for UBM approach.

Class	Samples by class	UBM-data samples	Training samples	Test samples
Attack	251712	125954	75337	50422
Benign	251712	125758	75690	50264

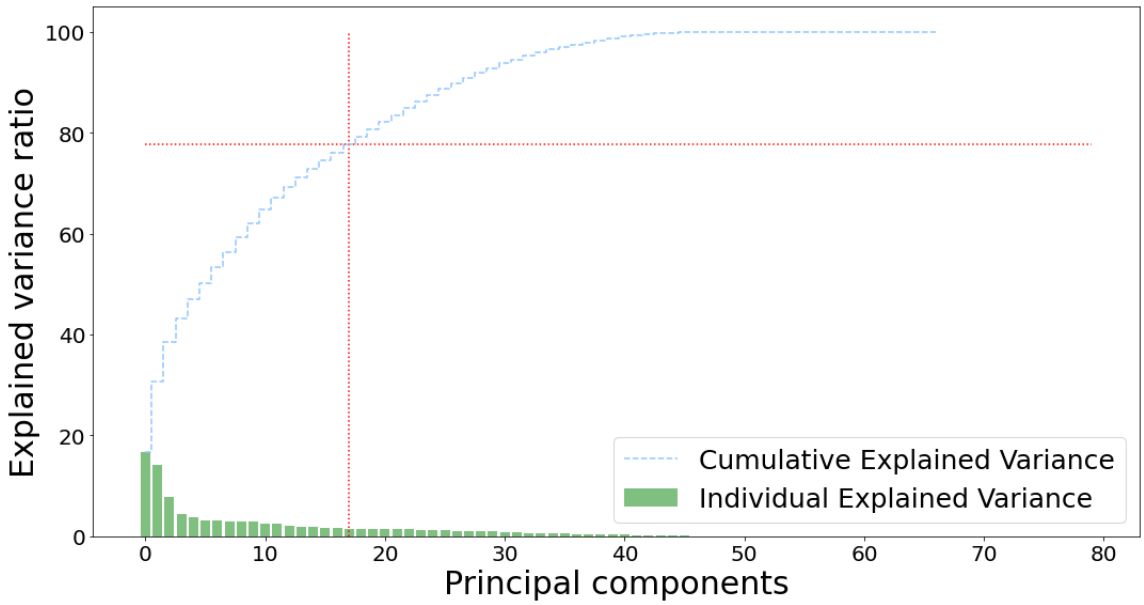


Fig. 6. Cumulative Explained Variance applying PCA.

For the GMM approach, 70% of the dataset is used for training and estimating the parameters of each combination of Gaussians, and 30% is used in the test step, as shown in Table 6. Both scenarios will rely on the scoring metric included in the sklearn library for classification tasks. The analysis was performed by testing with models that had from 1 to 19 Gaussians in order to compare the results of different combinations of distributions.

Table 6. Dataset segmentation for GMM approach.

Class	**Samples by class**	**Training samples**	**Test samples**
Attack	251712	201399	50313
Benign	251712	201340	50372

For the Random Forest experiment, the data is split in the same manner as the GMM experiment. In this case, a search grid is used to determine the best model, changing parameters like n estimators (number of trees in the model), max features (the number of features to consider when looking for the best split), max depth (the maximum depth of the tree), and criterion (the function to measure the quality of a split) [62], giving a total of 144 different models. These parameter will take values as follows:

- n_estimators = {15, 20, 25}
- max_features = {3, 5, 7, 9}
- max_depth = {None, 6, 7, 8, 9, 10}

- criterion = {"gini", "entropy"}

Taking into account the explanation about the three experimental scenarios; the following section shows the performance of each experiment and some conclusions about this work.

4.1.4 Performance Evaluation

This section shows the performance for each experiment, mentioning the advantages or shortcomings found in the implementation of each algorithm, starting with the UBM experiment, followed by the GMM experiment, and finishing with the RF approach. As preliminary conclusions, we found the RF approach worked very well in this classification problem showing better performance than GMM and UBM. GMM performed better than UBM. Finally, the following results show an important first step to use these novel techniques in the networking area.

UBM scenario For this experiment, the accuracy is shown in Table 7 and Fig. 7a. After 9 Gaussians overfitting over the attack class appears, whilst with less Gaussians, the accuracy goes from 60% to 77.5%. Table 8 shows a summary of this experiment, including the precision and sensitivity of the scenario.

Table 7. Accuracy by gaussian in UBM scenario. The accuracy average is 56.3%.

Gaussians	1	2	3	4	5
ACC	0.617	0.707	0.655	0.696	0.602
Gaussians	6	7	8	9	10
ACC	0.775	0.400	0.663	0.537	0.494
Gaussians	11	12	13	14	15
ACC	0.544	0.501	0.501	0.501	0.5001
Gaussians	16	17	18	19	
ACC	0.501	0.501	0.501	0.501	

Additionally, the Confusion matrix for 2, 3, 4, and 6 Gaussians are the best result found.

	Confusion Matrix - 2		Confusion Matrix - 3	
	Attack	Benign	Attack	Benign
Attack	46723	3698	21420	29001
Benign	25820	24444	5758	44496

	Confusion Matrix - 4		Confusion Matrix - 6	
	Attack	Benign	Attack	Benign
Attack	28914	21507	36447	13974
Benign	9106	41158	8666	41598

GMM scenario For this experiment, the accuracy is shown in Table 9 and Fig. 7c. This model shows a good classification of the test dataset in the different combinations

Table 8. Summary results for UBM.

Gaussian	**Precision %**		**sensitivity %**	
-	Attack	Benign	Attack	Benign
1	57	85	95	28
2	64	87	93	49
3	79	61	42	89
4	76	66	57	82
5	59	62	69	51
6	81	75	72	83
7	35	43	23	57
8	65	68	72	61
9	52	55	67	41
10	50	49	50	49
11	55	54	52	57
12	50	0	100	0
13	50	0	100	0
14	50	0	100	0
15	50	0	100	0
16	50	0	100	0
17	50	0	100	0
18	50	0	100	0
19	50	0	100	0

from 1 to 19 Gaussians; the optimal point in this experiment could be from 3 to 9 Gaussians, where the accuracy values are good and the model is not too complex or too simple to represent the phenomenon. Table 10 shows a summary of this experiment.

Table 9. Accuracy by gaussian in GMM scenario. The accuracy average is 80.3%.

Gaussians	1	2	3	4	5
ACC	0.605	0.735	0.770	0.771	0.769
Gaussians	6	7	8	9	10
ACC	0.814	0.826	0.828	0.823	0.833
Gaussians	11	12	13	14	15
ACC	0.823	0.825	0.824	0.828	0.828
Gaussians	16	17	18	19	
ACC	0.833	0.838	0.839	0.843	

The best confusion matrix for 6, 7, 8, and 9 Gaussians are:

Table 10. Summary results for GMM.

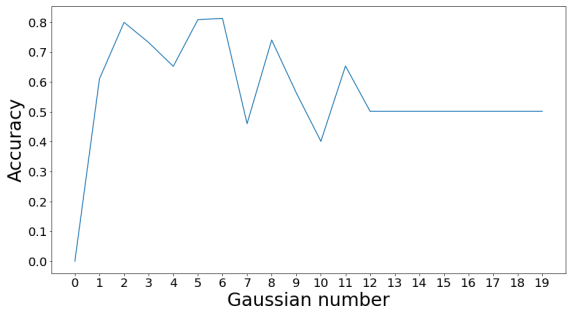
Gaussian	**Precision %**		**sensitivity %**	
-	Attack	Benign	Attack	Benign
1	56	83	95	26
2	77	71	68	79
3	92	70	59	95
4	93	70	59	95
5	93	70	58	95
6	90	76	70	92
7	94	76	70	96
8	94	76	70	95
9	92	76	70	94
10	95	76	70	97
11	93	76	70	95
12	94	76	70	95
13	94	76	70	95
14	95	69	76	96
15	95	69	76	96
16	95	69	76	96
17	96	77	70	97
18	96	77	70	97
19	97	77	71	98

	Confusion Matrix - 6		Confusion Matrix - 7	
	Attack	Benign	Attack	Benign
Attack	35355	14958	34997	15316
Benign	3785	46587	2201	48171

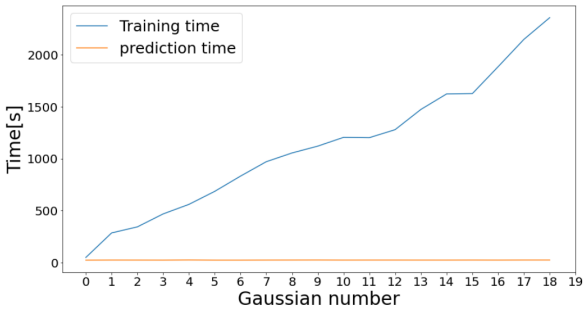
	Confusion Matrix - 8		Confusion Matrix - 9	
	Attack	Benign	Attack	Benign
Attack	35330	14983	35436	14877
Benign	2353	48019	2899	47473

Additionally, Figs. 7b and 7d show the training time and prediction time on each experiment, this can be useful to understand how much time is required to train and use a model with different number of Gaussians. Actually training time is relatively low, considering the amount of samples, having around 40-500 seconds for the UBM experiment and 200-450 for the GMM experiment.

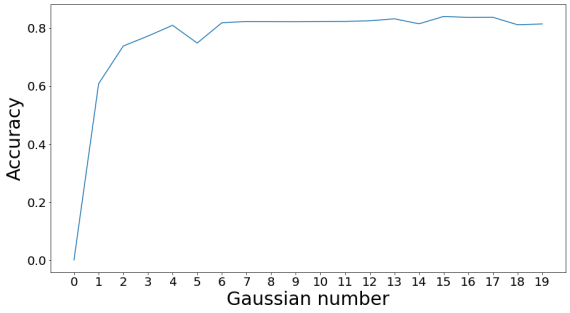
RF scenario As stated before, in this case, 144 models with a different set of parameters were evaluated. In Fig. 7e, the accuracy went from 85.13% up to 96.7% showing better results in comparison with the GMM and UBM scenarios. Table 11 shows 10 of 144 results where the second column is the accuracy of each result and the next



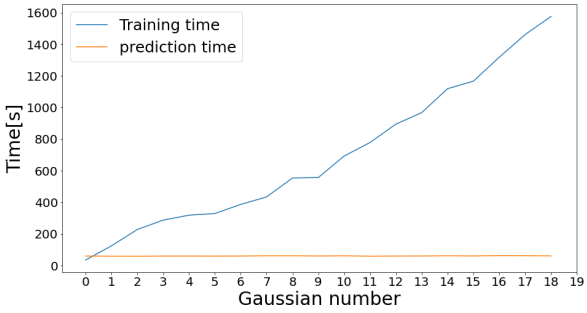
(a) Accuracy results for UBM scenario



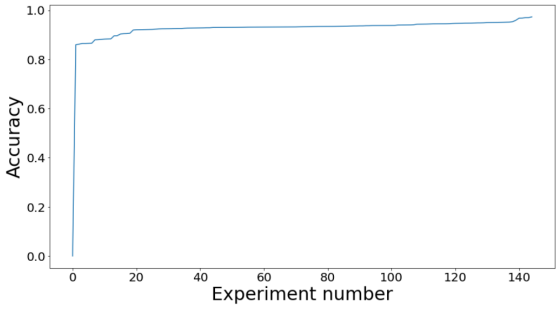
(b) Training time and prediction time for UBM scenario



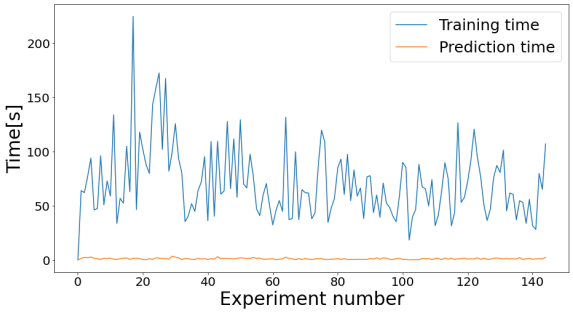
(c) Accuracy result for GMM scenario



(d) Training time and prediction time for GMM scenario



(e) Accuracy results for RF scenario



(f) Training time and prediction time for RF scenario

Fig. 7. Accuracies, Training time and Prediction time for experimental scenarios.

columns depict the chosen parameters to generate each model.

The training time by each model in Fig. 7f depends only on the parameters chosen. For the last three models, the training time is around 35-51 seconds, and the prediction time is around 0.79-1 seconds for the RF experiment.

The best three results have the following confusion matrices where the accuracy by each one is 96.1%, 96.2%, amd 96.7% respectively:

	Confusion Matrix - 17		Confusion Matrix - 18	
	Attack	Benign	Attack	Benign
Attack	45315	4998	46772	3541
Benign	196	50176	340	50032

	Confusion Matrix - 19	
	Attack	Benign
Attack	46788	3525
Benign	266	50106

The results of the experiments show that the RF algorithm works very well, obtaining up 96% of accuracy in this classification problem which is not that complicated and the simplicity of this problem may be one reason why the methods GMM and UBM did not work better than RF, but the first steps with the others two methods are really good because this shows an opportunity to improve the GMM and UBM methods and enrich the model to perform more complex classification experiments due to the GMM and UBM techniques worked very well in multiclass classification or multiclass verification problems in areas as speech recognition according to [14, 36]. These methods could show a better performance in problems where the goal is to classify different kinds of cyberattacks or classify the different strategies to deploy the same type of attack; there the robustness of the GMM or the UBM can be used. Additionally, these results are a good first approximation to improve the experiments and begin to work with these techniques in other domains. The GMM and UBM could work better in more complex scenarios, since these techniques are models that could generalize better a phenomenon. A further work could be a study that includes other databases including different features in order to compare the performance of the GMM and UBM against traditional ML techniques. In comparison, the training times for UBM and GMM are longer than the RF alternative, but these are short times taking into account that there are some ML models that can take much longer in the training process. As a future work, these training times could allow to make an automatic retraining process that could contribute to improve the performance of the GMM and UBM over time.

Note: Accuracy was used to show the main results and facilitate comparison of the scenarios presented in this chapter. In addition, the confusion matrix was used to expose the true positives, true negatives, false positives and false negatives to easily identify classification errors in each scenario. Also, the information from the confusion matrix allows the calculation of other metrics such as precision, recall and F1 score Eqs. 9,10,11 that are commonly used to analyze ML techniques.

Table 11. Summary for 19 results for RF.

N	**accuracy**	**criterion**	**max depth**	**max features**	**n estimators**
1	0.851	gini	10.0	9	20
2	0.852	gini	10.0	9	25
3	0.853	gini	6.0	9	15
4	0.853	gini	6.0	9	20
5	0.854	gini	6.0	9	25
6	0.854	gini	None	7	15
7	0.867	gini	9.0	7	15
8	0.867	gini	9.0	3	15
9	0.867	entropy	9.0	7	25
10	0.868	gini	10.0	5	15
11	0.869	entropy	None	3	25
12	0.872	entropy	10.0	7	25
13	0.936	entropy	9.0	9	20
14	0.937	entropy	9.0	7	15
15	0.948	gini	6.0	5	15
16	0.948	gini	6.0	5	20
17	0.961	gini	7.0	3	15
18	0.962	entropy	7.0	5	15
19	0.967	entropy	6.0	5	15

$$\boldsymbol{precision} = \frac{tp}{tp+fp} \tag{9}$$

$$\boldsymbol{recall} = \frac{tp}{tp+fn} \tag{10}$$

$$\boldsymbol{F1-score} = \frac{precision * recall}{precision + recall} \tag{11}$$

4.1.5 Preliminary conclusions for the proposed detection strategy

The UBM and GMM are powerful techniques that can be used in many different scenarios such as classification of security attacks in networking where the data to generate the models and probe the algorithms are cheap. The GMM technique provides better results in the classification for the two-class problem (BENIGN and DDoS), where for 8 Gaussians the obtained accuracy is 82.8% in comparison to the UBM technique where 6 Gaussians led to an accuracy of 77.5%. We found that with more Gaussians the model starts to show overfitting. The accuracy result for the UBM technique can be related to an insufficient capability to explain the phenomenon of the extracted

features used to generate the UBM. Is important to remark that the RF model performed better in terms of accuracy compared to the GMM and UBM, however, this work also tries to introduce techniques that work very well in other areas as speech recognition and implements them in the networking area, and more specifically in the cybersecurity area where the traditional techniques do not present relevant changes. These techniques could replace an Intrusion detection System or any other type of detection system in the future.

As future work, The GMM and UBM techniques could be implemented to classify more than two classes, e.g. trying to classify each kind of DoS/DDoS attack or trying to model a multi-class problem where the classes could be different kinds of cyberattacks. Additionally, a research could be conducted to find and construct other feature extraction processes that better describe the phenomenon, to improve the performance of the novel techniques implemented in this work.

Summarizing, this work uses the GMM technique as a attack detection strategy to introduce this novel ML technique and verify the effectiveness of it in a real NFV environment. This strategy consists to get the real time network traffic as input to perform the feature extraction based on Chapter 4.1.2, then, the processed data should pass through the GMM classifier based on Chapter 4.1.4 to determine if each flow traffic will be labeled as attack or benign traffic.

4.2 Attack mitigation process

This subsection will describe the mitigation process, also, this work will present an additional strategy to complement this process.

If any flow traffic is detected as attack, the mitigation strategy checks if a flow rule was already created for this malicious flow, in case the flow rule exists, the existing flow rules will drop the next flows that match these flow rules. But in case this flow rule does not exist, the strategy will send a request to SDN controller with the needed parameters to create the new flow rule to drop the next incoming traffic. After installing the new flow rule, each flow that matches with this flow rule will be dropped and this malicious traffic won't affect the performance of the NS.

If any flow traffic is classified as normal, the mitigation strategy focuses on determining if a high demand for the service is occurring to deploy a load balancing strategy and to guarantee service availability in order to ensure its optimal performance.

The load balancing strategy is implemented as a complement to tackle the availability

problems provoked by high volume of requests sent by the users. Here, the the strategy checks if the normal traffic throughput exceeds a certain threshold previously defined and also checks if the attack detection throughput is less than certain threshold previously defined to guarantee that the traffic throughput increase is not related to any attack. If this event occurs, the strategy performs a request to the NFVI to turn the second VNF on and makes that the LB works distributing the load in both VNFs. If this event does not occur, the strategy does not make anything and allows the normal traffic to reach the VNF.

5 Experimental settings

This section shows the experimental details of the implementation performed in this master project. Within these details we show:

- NFV platform: This subsection will explain the selected platform and its implementation as real NFV environment.
- Experimental scenarios: This subsection will explain each experimental scenario and their details
  - DDoS/DoS strategy: This will explain the implementation of the proposed DDoS/DoS detection and mitigation strategies.
  - Load balancing test: This will show how manage an undesired behaviors related to a web service.

5.1 NFV platform

Currently, there are different kind of NFV platforms that could be developed for different purposes, such as: to provide end-to-end services, to manage issues related to service placement, to focus on dynamic scaling, monitoring, among others. Also, these platforms could concentrate in providing an efficient infrastructural packet data path avoiding redundant processes [63]. Next, we present several categories of NFV platforms:

- **End-to-End service provisioning**: Oriented to service providers or network operators in order to provide end-to-end services. Some of these platforms are *UNIFY* [64], *Cloud4NFV* [65], *CloudBand* [66], *GNF* [67], *DeepNFV* [68], *NetFATE* [69], *SONATA* [70], among others.
- **VNF development**: They are dedicated to ease the development of VNFs and guarantee efficient VNF execution. Some of these platforms are *xOMB* [71], *CoMb* [72], *FlowOS* [73] *NetBrics* [74], *Scylla* [75], *LibNFV* [76], *Flick* [77], *ClickNF* [78], among others.
- **Holistic MANO system**: These kinds of platforms focused on creating completed MANO systems (they try to tackle all facet of MANO, such as the scheduling, monitoring, scaling, load-balancing, failover, and VNFs management). Some of them are *ETSO* [79], *OpenMANO* [80], *Open Baton* [81], *vConductor* [82], *T-NOVA* [83], among others.
- **Focused on a specific feature of the MANO framework**:

  - **Scaling and failover**: *Split/Merge* [84], *TFM* [85], *OpenNF* [86], *DiST* [87], *LEGO* [88], among others.
  - **Scheduling**:*NFVnice* [89], *EdgeMiner* [90], *ResQ* [91], *NetContainer* [92], among others.
  - **Profiling**: *NFV-vital* [89], *Gym* [93], *ConMon* [94], *Symperf* [95], *KOMon* [96], among others.
  - **Secure execution**: *vEPC-sec* [97], *SplitBox* [98], *Embark* [99], *BSec-NFVO* [100], *S-NFV* [101], among others.
- **NFVI Acceleration**: These platforms are focused on delivering services without compromising performance. Some of these platforms are *NetVM*[102], *OpenNetVM*, *NetML*, *ClickOS*, *P4SC*, *P4NFV*, *OpenANFV*, *UNO*, among others.

There are several platforms to implement an NFV environment with different purposes, depending on the project goal is possible to use a specific platform or a general-purpose one.

In our case, OpenMANO (also called OSM) was selected as it implements the ETSI NFV MANO framework and also guarantees different performance levels. OSM also allows to use different VIM platforms such as OpenVim, OpenStack, AWS, VMware's vCloud Director, among others. Also, it has a graphical user interface providing easier management. OSM could be used with SDN in order to easily program paths and traffic engineering in the network [63].

OSM is part of holistic MANO systems and is an open source project that can be used without license restrictions. The way OSM talks with the VIMs and VNFs is shown in Fig. 8.

In first place, OSM talks to the VIM for the deployment of VNFs and virtual links (VLs) by connecting them. Next, OSM talks to VIM to deploy a VNF or set of VNFs.

In order for OSM to work, it is assumed that:

- Each VIM has an API endpoint reachable from OSM.
- Each VIM has a so called network management which provides IP addresses to VNFs.
- That network management is reachable from OSM.

OSM's user guide [103] works with OpenVim or Openstack [104] as a open source

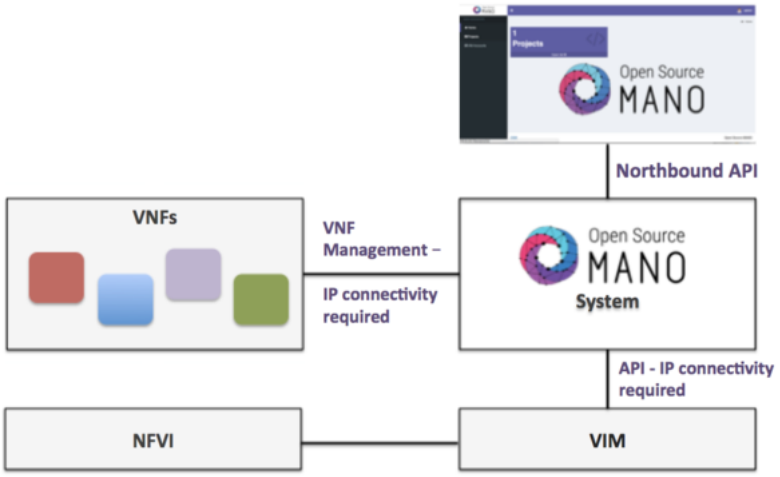


Fig. 8. OSM interaction with VIM and VNF.

VIM alternatives. it was selected as VIM taking into account that OpenStack is a huge VIM platform with many different types of components used in tons of private cloud projects, it is OS supported by Canonical, and it counts with a large community to provide technical support.

5.2 Experimental setup requirements

To build and configure the experimental environment, the minimum requirements on each server or one work station are shown in Table 12. Fig. 9 shows the experimental environment with its different components.

Table 12. Minimum devices requrements.

Device	# Cores	RAM	Disk	Net Interfaces	OS
OSM	2 CPUs	8 GB	40 GB	1-LAN 1-Internet	Ubuntu18.04 (64-bit variant required)
OpenStack	4 CPUs	8 GB	500 GB	1-LAN 1-Internet	Ubuntu18.04 (64-bit variant required)
RYU	1 CPU	4 GB	30 GB	1-LAN	Ubuntu16.04 LTS or later
Attacker	4 CPUs	12 GB	30 GB	1-Internet	Kali 2020.1

The architecture in Fig. 9 consists of the following modules: OSM, OpenStack, SDN controller and the attacker.

- OSM is responsible for defining and deploying the NSs or VNFs through the VIM

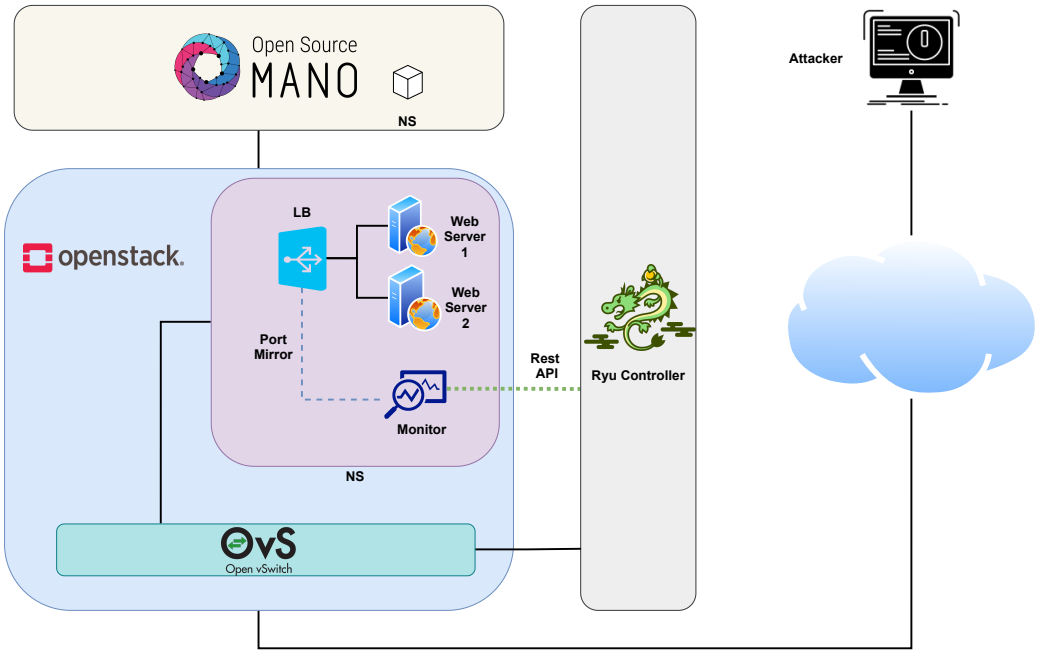


Fig. 9. Experimental environment architecture.

and to guarantee their optimal operation.

- Openstack ensures that the VM, virtual link, public IPs, policy groups, or any other component that conforms the NS or VNF have the proper functioning according to the policies, parameters, and resources defined by OSM.
- The SDN controller provides an easy management of network policies and it allows to run new flow rules. It is also in charge of defining a custom network behavior of our NS or components deployed inside OpenStack.
- The network operation defined by the SDN controller is only possible thanks to the Open Virtual Switch (OVS) inside Openstack that allows to specify every network action performed by each component on the NS.
- The attacker is focused on deploying the DDoS/DoS attacks towards the WEB servers.

The experimental environment in Fig. 9 works as follows:

- The administrator creates the NS description file (YAML) and the VNF description files (YAML) according to OSM's documentation to define how the NS will be created, what kind of VNFs will be deployed inside it, and how they will work together.
- Those NSs and VNFs description files will be uploaded on the OSM platform and

then, they could be used to deploy the real NSs and VNFs on the VIM (Openstack).

- After deploying the NS in OSM and OpenStack, its VNFs could be reachable through SSH or WEB interfaces if the security policies were correctly defined.

The NS implemented in this work (see Fig. 10) consists of two web servers accessible through port 80 and one monitoring server to deploy the detection and mitigation strategy. Moreover, it is necessary to make some additional configurations inside the VIM in order to monitor the network traffic, and to deploy a load balancer on-demand as an additional feature for this work. For this reason, we implement a port mirror to the web server one to other VNF that performs the task of monitor server, and a load balancer between the web servers one and two.

The NS was deployed using OSM and Openstack. By default, and according to the network security policies defined inside Openstack, a set of rules were set up on the OVS to guarantee the communication inside the NS and to make the VNFs reachable from an external point (Internet) or the LAN. Additionally, the RYU SDN controller sets new Openflow rules inside the OVS performing custom network management or creating SDN applications to add new features to the network.

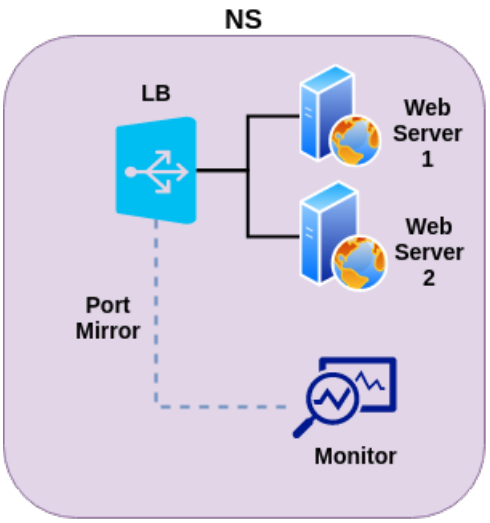


Fig. 10. Network service deployed through OSM and OpenStack.

The attack deployed in the experimental scenarios were performed by retransmitting the traffic captures obtained in the testbed **"Intrusion Detection Evaluation Dataset (CICIDS2017)"**, so it is not possible to know the configuration parameters that were used them and it is a similar situation to what happens in the real life.

5.3 Attack detection process

Fig. 11 represents the detailed explanation the attack detection process and attack mitigation process (in the next section 5.4) as a flow chart to make easier to follow all the strategies.

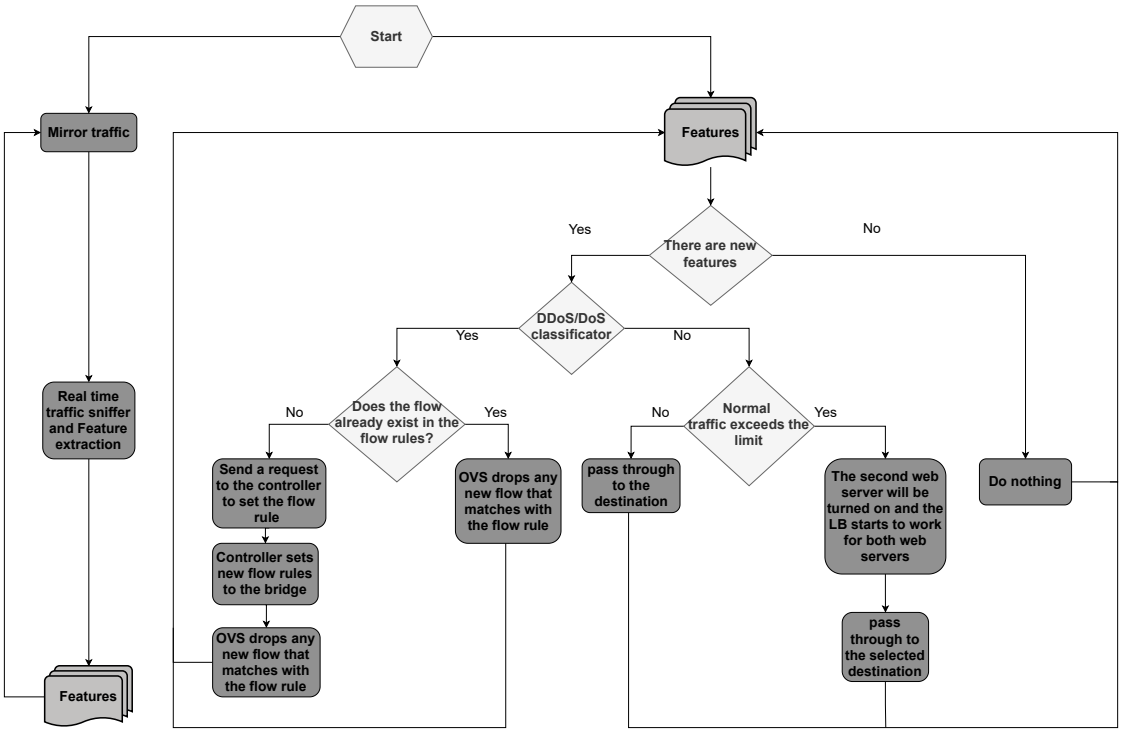


Fig. 11. Flowchart for the developed strategy.

This implementation runs in two threads: The first one is focused on taking the mirrored traffic as an input and performing the feature extraction process. This process is performed over a window of 60 seconds and it calculates 85 features; after eliminating features that do not provide information such as src_ip, src_port, dst_ip, dst_port, among others, 77 out of these 85 features are used in the next steps. Table 13 shows some of those features.

The feature extraction process is implemented using a library called CICFlowMeter [52, 53], developed by the Canadian institute for cybersecurity of the New Brunswick university. Each entry on the feature extraction file represents a traffic flow in a window of 60 seconds, and this entry shows different statistical metrics for this detected flow. After the feature extraction process, this extracted data will be saved to perform the detection strategy.

Table 13. Features generated by the algorithm tool.

Feature Name	Description
Flow Duration	Duration of the flow in Microsecond
Total FWwd Packet	Total packets in the forward direction
Total Bwd packets	Total packets in the backward direction
Total Length of Fwd Packet	Total size of a packet in forward direction
Total Length of Bwd Packet	Total size of the packet in backward direction
Fwd Packet Length Min	Minimum size of the packet in forward direction
...	...
std idle	Standard deviation time a flow was idle before becoming active
Init Win bytes forward	The total number of bytes sent in the initial window in the forward direction
Init Win bytes backward	The total number of bytes sent in the initial window in the backward direction
Act data pkt forward	Count of packets with at least 1 byte of TCP data payload in the forward direction

The second part of this algorithm consists of reading each new feature extraction file to pass it through the classifier in order to determine if each entry inside the feature extraction file is normal traffic or an attack.

This process is based on the trained GMM-based technique to detect a DDoS/DoS attack explained in Chapter 4.1.

5.4 Attack mitigation process

If the detection process outputs a **Yes** (attack), the algorithm checks if a flow rule was already created for this malicious flow, in case the flow rule exists, the existing flow rules in the OpenFlow switch will drop traffic belonging to the attack. But in case this flow rule does not exist, the algorithm will send a request to the RYU API controller with the needed parameters (source IP, destination IP) to create the new flow rule and

set it to the right OpenFlow switch. After installing the new flow rule, each flow that matches with this flow rule will be dropped and this malicious traffic won't affect the performance of the NS.

If the output of the detection process is **No**, the algorithm in this section of the flowchart focuses on determining if a high demand for the service is occurring to deploy a load balancing strategy and to guarantee service availability in order to ensure its optimal performance. Here, the algorithm checks if the normal traffic throughput exceeds a certain threshold defined in the setup parameters and also checks if the attack detection throughput is less than certain threshold to guarantee that this traffic throughput increase is not related to any attack. If the answer is **Yes**, the algorithm performs a request to the **NOVA** component of OpenStack to turn the second VNF on and makes that the LB works distributing the load in both VNFs. If the answer is **No**, the algorithm does not make anything and allows the normal traffic to reach the VNF.

The Classifier used in this algorithm was explored and detailed in Chapter 4. It implements the GMM technique with 9 Gaussians after performs a preprocessing of the data, The database used to generate the classification model was **"Intrusion Detection Evaluation Dataset (CICIDS2017)"**, it contains benign network traffic and updated common security cyberattacks such as Hulk, Slowloris, SlowHTTPTest, among others. The authors ensure that the dataset resembles the true real-world data, and CICFlowMeter was used to extract data from traffic flows [59]. They added 85 statistical features, but more can be included [60, 61].

5.5 Experimental scenarios metrics

This work uses different metrics to measure the web server performance:

- Transmitted packets.
- Received packets.
- Transmitted packets per second.
- Received packets per second.
- CPU consumption.
- Memory consumption.

For each specific scenario there are other metrics used like:

- **Attack mitigation scenario**:
  - Dropped flow rules installed in the OpenFlow switch table.
  - Number of packets dropped by the flow rules.
- **Load balancing scenario**:
  - web server latency.
  - web server status.

Those metrics help the comparison process on each experimental scenario and allow us to validate the effectiveness of the proposed strategies.

5.6 Attack detection and mitigation scenario

This section explains the detection and mitigation strategies based on the experimental environment shown in Chapter 5.1 and takes into account the aforementioned algorithm. This scenario only uses one of the web servers deployed in the NS and the monitor server. First, we perform an experiment without using the algorithm to detect and mitigate the DDoS/DoS attacks to use these results as a baseline to compare with the second experiment when the detection and mitigation strategies are enabled. The main goal with this scenario will be to perform a comparison where the effectiveness of the proposed strategy implementing this novel detection technique in network security is shown. Additionally, this opens the possibilities to test new ML techniques widely used in other engineering areas.

This attack is conformed by different techniques to generate a DDoS or DoS aimed to test the detection and mitigation strategies and their capacity to cover a wide area of these attacks. The used variations of DoS attacks are:

- **DoS Slowloris**: This tool performs the DoS attack “holds connections open by sending partial HTTP requests. It continues to send subsequent headers at regular intervals to keep the sockets from closing" [105].
- **DoS Slowhttptest**: “It implements most common low-bandwidth Application Layer DoS attacks, such as slowloris, Slow HTTP POST, Slow Read attack (based on TCP persist timer exploit) by draining concurrent connections pool, as well as Apache Range Header attack by causing very significant memory and CPU usage on the server." [106].
- **DoS Hulk**: “It is designed to generate volumes of unique and obfuscated traffic at a web server, bypassing caching engines and therefore hitting the server’s direct resource pool" [107, 108].

- **DoS GoldenEye**: “It is a python script that is meant for testing HTTP denial of service conditions leveraging HTTP Keep-Alive and NoCache" [109, 110].

The following is the workflow that will be used when the detection and mitigation strategies are being used and when they are not used:

Performing the DDoS/DoS attacks without the detection and mitigation strategy.

1. Test the connectivity scenario to guarantee the attacker reach the web server to make possible to deploy the attack.
2. Wait for a short time to store metrics without attack effects.
3. The attacker will start the attack to the web server.
4. The web server captures the metrics to evaluate the performance before and after the attack.

Performing the DDoS/DoS attacks using the detection and mitigation strategy.

1. Test the connectivity scenario to guarantee that the attacker reaches the web server to deploy the attack.
2. Start the detection and mitigation strategies as a service inside the monitor server.
3. Wait for a short time to store metrics without attack effects.
4. The attacker will start the attack to the web server.
5. The web server captures the metrics to evaluate the performance during the attack.

5.7 Load balancing scenario

This section explains the load balancing scenario added to the algorithm mentioned in Section 4.2 to show the way to implement strategies to figure out different situations that could appear in the life cycle of your NS. Specifically in this case, a high demand service not due to an external attack. To add this complementary feature to this work, it is required to implement a load balancer to the VIM for two web servers. Also, it is required to mirror the input/output traffic from this load balancer to the monitor server as shown in Fig. 9.

With this implementation ready, if for any reason an increment of throughput is experienced and it exceeds the previously defined threshold, and if there is no attack in progress, the algorithm will start the second web server to distribute the load that

could affect the principal web server and guarantees the availability, performance, and accessibility of this service.

This scenario is composed of two experiments. The first experiment will evaluate how the performance of the principal web server can be affected without the algorithm actions. The second experiment will have the monitor server running the algorithm to perform the load balancing action if the throughput exceeds the threshold and makes it possible to compare the benefits of this strategy with the first experiment. The workflow that will be used for both experiments is:

Performing the first experiment without the load balancing strategy.

1. Test the connectivity scenario to guarantee that the clients reach the web server to make it possible to deploy a high number of requests to it.
2. Wait for a short time to store metrics without high request effects.
3. Start the high number of requests using the tool called Apache-JMeter “designed to load test functional behavior and measure performance" [111].
4. The web server captures the metrics to evaluate the performance before and after the load test.

Performing the second experiment with the load balancing strategy.

1. Test the connectivity scenario to guarantee that the clients reach the web server to make it possible to deploy a high number of requests to it.
2. Start the load balancing strategy as a service inside the monitor server.
3. Wait for a short time to store metrics without high request effects.
4. Start the high number of requests using the tool called Apache-JMeter “designed to load test functional behavior and measure performance" [111].
5. The web server captures the metrics to evaluate the performance before and after the load test.

The source code of this implementation can be found on GitHub [112].

6 Performance Evaluation

This section shows the obtained results through deploying and capturing data in the experimental scenarios that were presented in the previous chapter for the DDoS/DoS attack and the load balancing test.

6.1 DDoS/DoS detection and mitigation strategies using GMM approach

This subsection explains the obtained results after deploying the DDoS/DoS attack through the first experiment and it will show what happens if we are not using the proposed detection and mitigation strategy and the benefits when we are using it.

As a first part of this experiment, we deploy a DDoS/DoS attack conformed by different techniques such as "DoS Slowloris", "DoS Slowhttptest", "DoS Hulk", "DoS GoldenEye" from the attacker to the Web server 1 in the experimental environment shown in Fig. 9. It is very important to remark that the most critical point during the experiment was around at 00:14 hours until 01:14 hours where the principal part of the DDoS/DoS attack had been deployed.

Figs. 12 and 13 depict the CPU and RAM behavior (Y-axis corresponds to the percentage of usage and X-axis corresponds to the day and time of the test) of Web server 1 with the detection and mitigation strategy disabled and enabled respectively; these Figures show that no anomaly is detected even knowing that this kind of attack sends thousands or millions of requests to the Web server. There are few samples that show a high CPU usage but they do not mean that the deployed attack is affecting the process capability of the Web server. The reason of this behavior is because they are designed to avoid keep opening the http session, and just obfuscate the Web server and kill the service during the attack.

Also, Figs. 14 and 15 show the amount of Tx packets, Rx packets, Tx pkt/s, and Rx pkt/s obtained with the detection and mitigation strategy disabled and enabled respectively. In Fig. 14 we can see two important events in the first section. The first one is the blue line that corresponds to the amount of Rx packets, and shows an increase of the incoming packets at the Web server in a short time. It means that in the first part of the experiment (around 3 hours), the Web server is being bombed by the major part of the DDoS/DoS attack. On the other hand in the second event, the green line depicts the Rx pkt/s in this same time interval that take high values in comparison with the

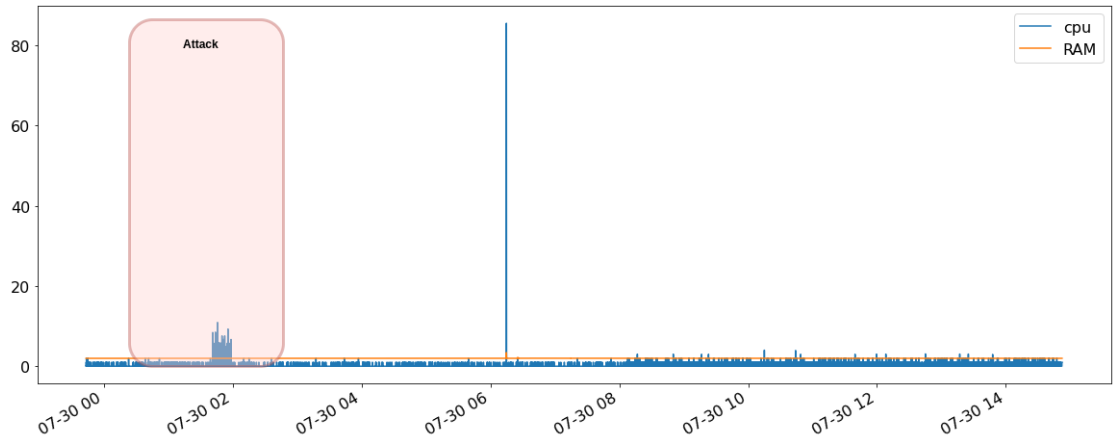


Fig. 12. CPU consumption of the Web server N°1 with the detection and mitigation strategy disabled.

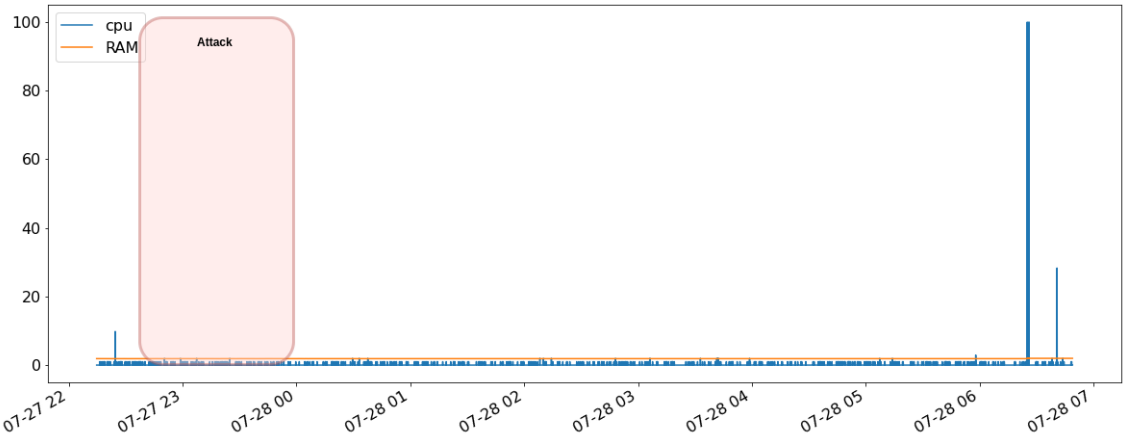


Fig. 13. CPU and RAM consumption of the Web server N°1 with the detection and mitigation strategy enabled.

rest of the experiment and follows a similar behavior to the blue line. Also, In Fig. 15 we can see an important decrease in the number of packets that arrive at the Web server. In first place, around 00:00 hours, a rise of the Rx packets was stopped, then, normal growth of the RX packets is noticed. In second place, around 01:00 hours, a second huge attack was stopped to allow the normal growth of the Rx packages that arrive at the Web server.

To be more precise, Table 6 shows each flow rule triggered by the mitigation strategy and installed by the SDN controller, a timestamp where each flow rule were installed, its lifetime, and the number of packets dropped by each flow rule.

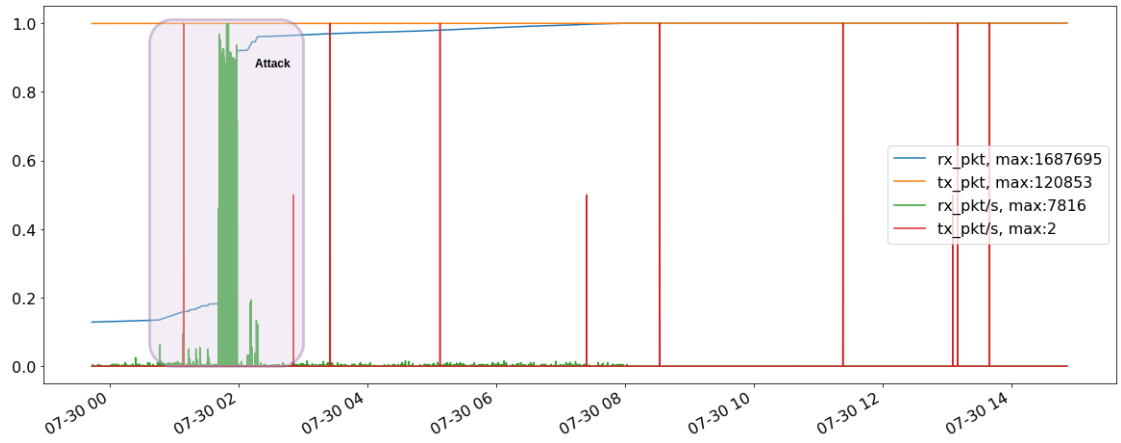


Fig. 14. Traffic statistics of the Web server N°1 with the detection and mitigation strategy disabled.

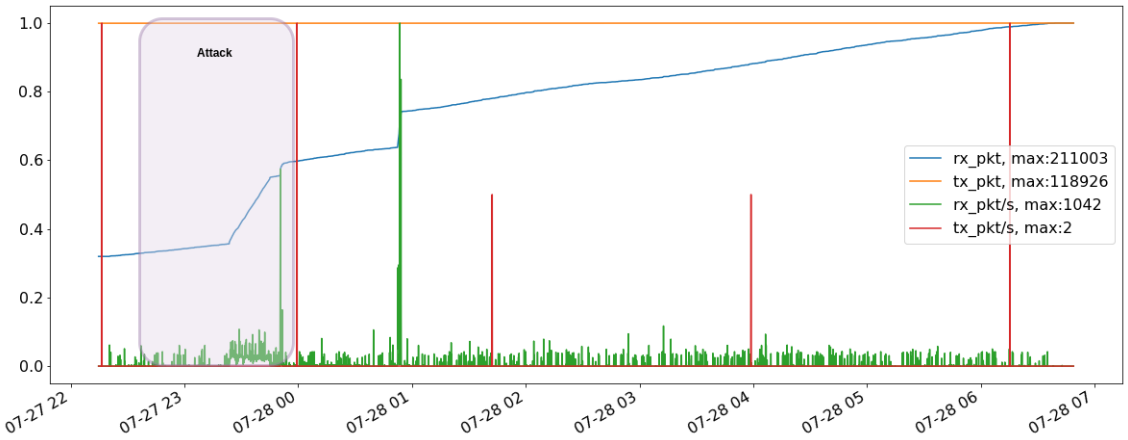


Fig. 15. Traffic statistics of the Web server N°1 with the detection and mitigation strategy enabled.

Table 14 depicts the flow rules installed by the proposal strategies on the OVS and shows that each sample was taken with 30 minutes of difference, also the number of packets dropped in this experiment implementing the detection and mitigation strategy is 1'316.065. This result shows that our strategy is effective in cases related to stopping an attacker that implements DDoS/DoS techniques and is important to remark that our strategy is agnostic to the src, dst IP, and ports due to these features do not provide information about the network phenomenon according to the the GMM model trained in Chapter 4. Also, It is very important to notice that in Fig. 14, the maximum amount of Rx packets received by the Web server were 1'470.443 packets (disabled strategy) and Fig. 15 shows that the number of packets that arrived at the Web server was 143.545 (enabled strategy) that is much lower than the ones of the

first test, which means that the detection and mitigation strategy is blocking around to 89.5% of the incoming packets that arrive at the Web server with a 0.96% of error, corresponding to the flows rules with src IPs [192.168.10.3, 192.168.10.19] and dst IP 192.168.100.112; these flow rules are false positive flows detected as attacks but they correspond to around 1% of whole network traffic analyzed in this experiment.

As a review of the experiment, Figs. 14 and 15 show the difference that exists if we implement the developed strategies to detect and mitigate the selected attacks maintaining the Web service available for real requests sent by customers and also, preventing external attacks. This experiment shows the effectiveness of the developed strategy to detect the DDoS/DoS attacks in three relevant moments (row 8, 9, and 10 according to Table 14) where 1'301.967 packets or the 90% of the traffic sent by the external attacker was dropped, and just allows the real request sent by the customer to pass through to the Web servers (10% of the total traffic).

In conclusion, the developed strategy covers the malicious attacker detecting and mitigating the selected attacks, and also, guarantee an optimal service offered by this Web server and prove the effectiveness of the proposed detection strategy based on GMM technique in a real NFV environment. The other way to affect the service availability of the exposed Web server will be covered by the next strategy and it occurs when a lot of customers send requests to the Web server.

6.2 Load balancing strategy

This subsection explains the obtained results after deploying a high throughput test through the second experiment and it also shows what happens if we are not using the load balancing strategy and what is the benefit of using it. This strategy is an additional feature to cover the the other side of Denial of service where we have a huge amount of requests that are not triggered by an external attacker and correspond to regular traffic sent by a lot of customers that are using the Web service.

As first part of this experiment we implemented a well-known tool called Apache Jmeter that is developed to perform a load test of our application and designed to evaluate its performance. Additionally, to test this strategy created two profiles were implemented, one of them to test the Web server 1 directly and the other one to test the strategy with the load balancer as the target. With this in mind, we used a base profile to deploy the performance test where we defined 15.000 threats simulating the number of users for the Web server, as a ramp-up period we used 0.5 seconds and, after that, we used a GET request to the respective dst IP according to the target. In

Table 14. Flow rules installed in the OVS inside of OpenStack.

N	start	duration (min)	src IP	dst IP	# Dropped pkts
1	2021-07-27 21:05:47	-	-	-	-
2	2021-07-27 22:14:42	-	-	-	-
3	2021-07-27 22:44:42	20.4	192.168.10.3	192.168.100.112	661
4	2021-07-27 22:44:42	7.4	192.168.10.19	192.168.100.112	0
5	2021-07-27 23:14:42	50.4	192.168.10.3	192.168.100.112	1679
6	2021-07-27 23:14:42	37.4	192.168.10.19	192.168.100.112	278
7	2021-07-27 23:44:42	-	-	-	-
8	2021-07-28 00:14:42	22.3	172.16.0.1	192.168.100.112	25356
9	2021-07-28 00:44:42	52.3	172.16.0.1	192.168.100.112	1272308
10	2021-07-28 01:14:42	20.3	172.16.0.1	192.168.100.112	4303
11	2021-07-28 01:14:42	5.4	192.168.10.3	192.168.100.112	374
12	2021-07-28 01:44:42	35.4	192.168.10.3	192.168.100.112	1472
13	2021-07-28 02:14:42	4.4	192.168.10.3	192.168.100.112	254
14	2021-07-28 02:44:42	34.4	192.168.10.3	192.168.100.112	1182
15	2021-07-28 03:14:42	-	-	-	-
16	2021-07-28 03:44:42	26.4	192.168.10.3	192.168.100.112	875
17	2021-07-28 04:14:42	56.4	192.168.10.3	192.168.100.112	2173
18	2021-07-28 04:44:42	20.4	192.168.10.3	192.168.100.112	793
19	2021-07-28 05:14:42	50.4	192.168.10.3	192.168.100.112	1785
20	2021-07-28 05:44:42	19.4	192.168.10.3	192.168.100.112	558
21	2021-07-28 06:14:42	49.4	192.168.10.3	192.168.100.112	1550
22	2021-07-28 06:44:42	18.3	192.168.10.3	192.168.100.112	464
Total	-	-	-	-	1'316.065

this case, the target is the Web server directly.

For the second part of this experiment, we deployed a similar experimental setting as in the previous one, just changing the Web server IP to the load balancer IP as the target.

Figs. 16 and 17 depict the CPU and RAM behavior (Y-axis corresponds to the percentage of usage and X-axis corresponds to the day and time of the test) of Web server 1 with the load balancing straategy disabled and enabled respectively; here we can see how the CPU processes the http requests sent to the Web server by the customers during the test and the expected measure about RAM consumption.

Fig. 16 represents what happens when the Web server processes all requests by itself and we can see that there are moments where the Web server stopped to process request due to its impossibility to handle them. However in Fig. 16, we can see that the CPU usage is much less than the previous test, and also, we did not see any stop processing the incoming request. Regarding to the RAM usage, we could not see any relevant consumption for both tests.

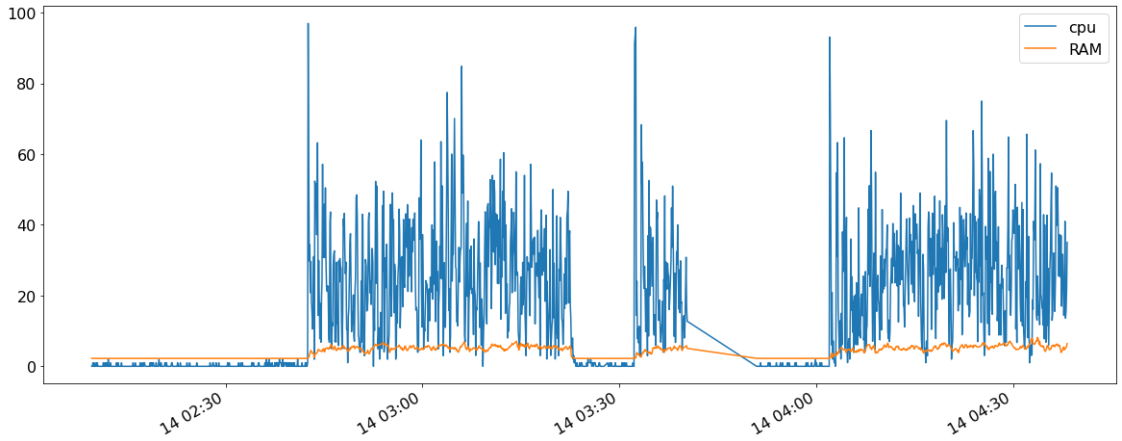


Fig. 16. CPU and RAM consumption of the Web server N°1 with the load balancing strategy disabled.

Figs. 18 and 19 show the amount of Tx packets, Rx packets, Tx pkt/s, and Rx pkt/s (each metric in these chars has been normalized with each max value, and the X-axis represents the test time) obtained during each experiment. In Fig. 18 we found that the Rx ptk/s and Tx pkt/s manage a huge amount of pkt/s with values around 25.690 and 18.548 respectively, additionally, it is important to notice that the maximum

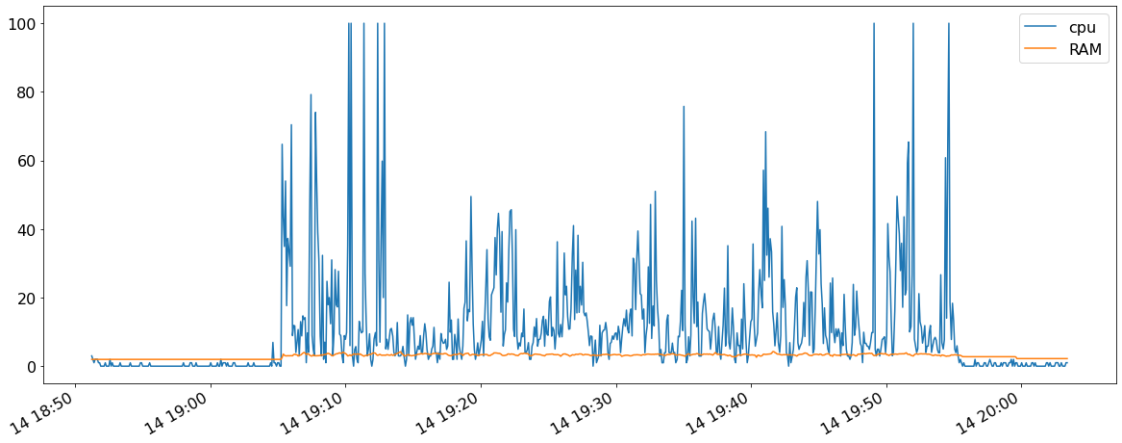


Fig. 17. CPU and RAM consumption of the Web server N°1 with the load balancing strategy enabled.

amount of incoming packets is equal to 27'651.899 and in case of the transmitted packets the maximum value of it is 16'150.320. Regarding Fig. 19, we found that the Rx and Tx packets and the Rx ptk/s and Tx pkt/s have different behavior in comparison with the above test where the density of Rx and Tx pkt/s is much less with 11.775 and 5.469 pkt/s respectively, and the amount of Rx and Tx packets were 7'381.435 and 2'114.592 respectively. We can see at the moment a huge difference when enabling the LB strategy and how it benefits the NFV environment and our Web service comparing the measures obtained in both experiments.

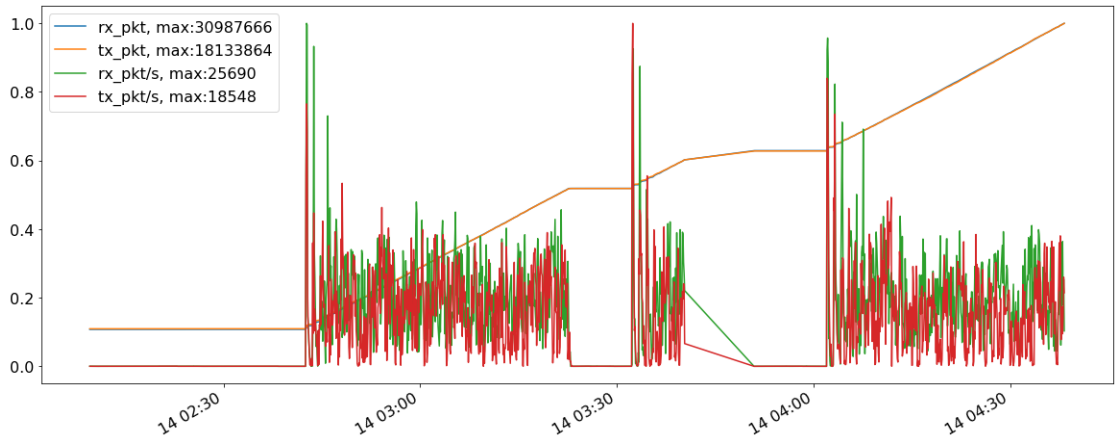


Fig. 18. Traffic statistics of the Web server N°1 with the load balancing strategy disabled.

On the other hand, Figs. 20 and 21 show the Web server status (it makes reference to where the Web server is online or offline) where the load balancing strategy is disabled and enabled respectively. Figs. 20, and 21 just have two possible values [0 -

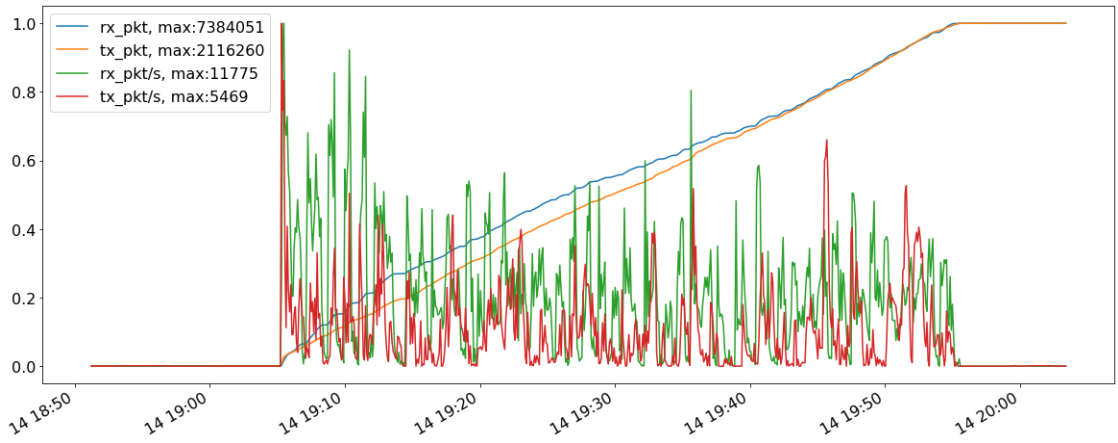


Fig. 19. Traffic statistics of the Web server N°1 with the load balancing strategy enabled.

1] to represent the Web server status, related to Fig. 20 we can see here that there are five special moments (between [17:00 - 18:00], [19:00 - 23:50], [00:00 - 02:45], [03:25 - 03:30], and [03:45 - 04:00]) during the experiment where this metric could not be measured, but in case of Fig. 21 we could not see an interruption of service.

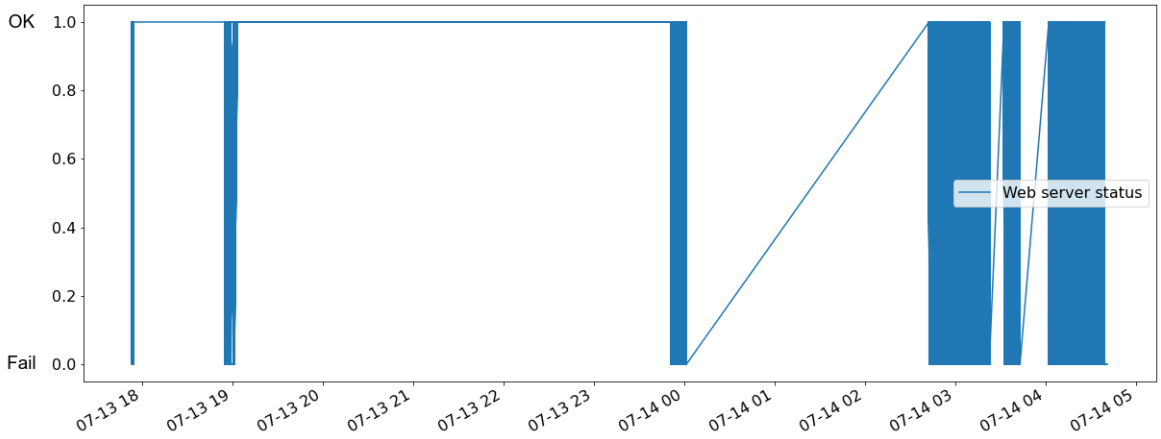


Fig. 20. Web server status with the load balancing strategy disabled.

Also, Figs. 22 and 23 show the Web server latency along both test. Regarding Fig. 22, it was also not possible to measure the latency for the Web server for the special situations mentioned for the Web server status where the load balancing strategy was disabled. It is important to remark that the latency in this test takes values near to 150.000 ms where the Web server can replay the requests, and it represents a huge time to wait for an answer by the Web server. Fig. 23 shows the latency measure for the Web server using the load balancing strategy, the measure for this metric is remarkable because in this case, the Web server latency (55.000 ms on average) in this case, is

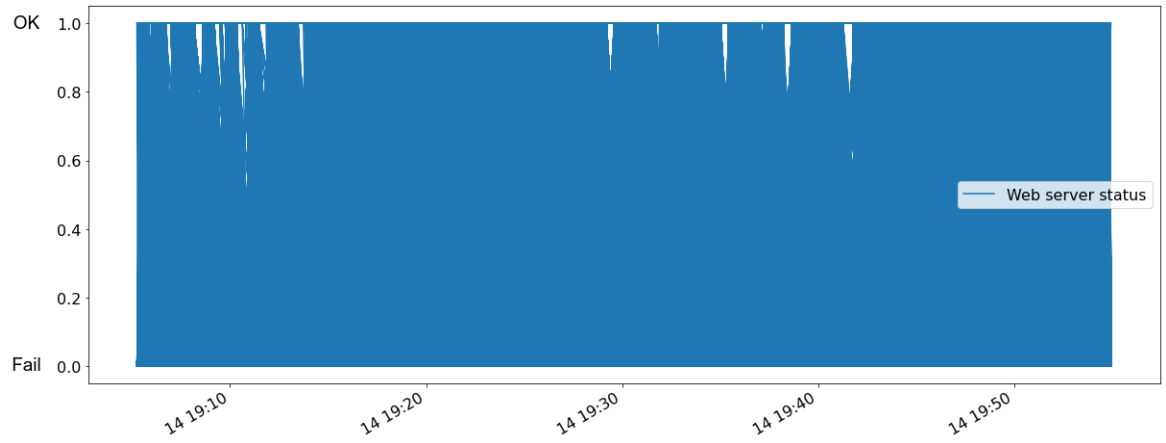


Fig. 21. Web server status with the load balancing strategy enabled.

three times smaller than the latency obtained in the previous test (150.000 ms). This improve represents around 36.6% of better behavior handle the thousands of request using the load balancing strategy.

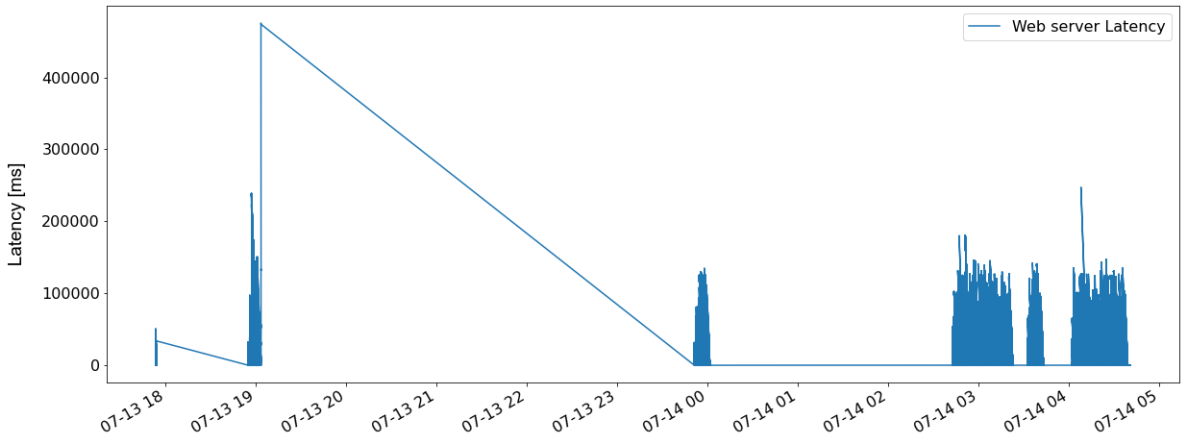


Fig. 22. Latency measure for the Web server N°1 with the load balancing strategy disabled.

These results show the effectiveness of the load balancing strategy to cover and manage situations with a huge volume of traffic requests sent to a Web server and proving that it is capable to preserve the service availability and the benefit of using it was over 36% much efficient in contrast to not using it. To summarize, Table 15 shows a comparison metrics capture for both experiments where we can find differences about the amount of Rx and Tx packets processed by the Web server, the remarkable decrease of the Web server latency, and also, that the CPU consumption is much less due that the request sent to the load balancer was distributed through the pool of Web servers that we configure for our service (in this case there were two).

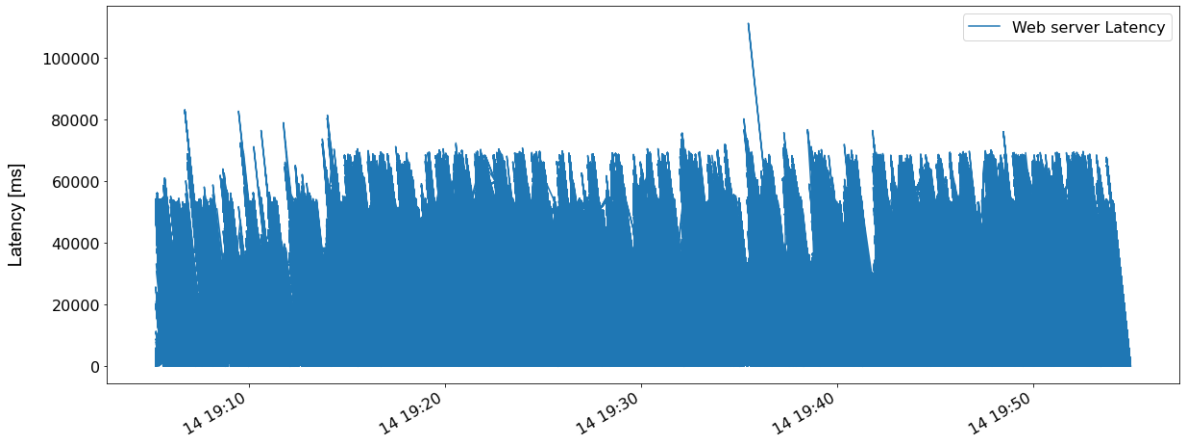


Fig. 23. Latency measure for the Web server N°1 with the load balancing strategy enabled.

Table 15. Summarize the Load balancing strategy.

——	**LB disabled**	**LB enabled**
Rx pkts	27'651.899	7'381.435
Tx pkts	16'150.320	2'114.592
Rx pkt/s	25.690	11.775
Tx pkt/s	18.548	5.469
Latency (ms)	150.000	55.000

The experiment where the load balancing strategy was enabled shows a huge benefit handling millions of requests from customers that are using the Web server, also, if the pool of web servers were larger than the ones we used in this experiment, we could be able to handle much more customers and make our Web service more robust without concern about the Web server performance.

To summarize, the load balancing strategy is an extension of the DDoS/DoS detection and mitigation strategy, because it covers the other side of the denial service caused by a lot of real customers that are using the Web service exposed in the NFV environment and the DDoS/DoS strategies cloud not be able to cover. For this reason, the strategies that have been shown in Sections 6.1 and 6.2 take an important role to guarantee the service available for users even if it does not matter what is affecting the exposed service.

7 Conclusions and Future work

During the literature review, we identified that the most critical point for this novel NFV architecture is the NFVI layer where we found that major part of the attack vectors used in the NFVI layer also affect the other NFV layers. Additionally, this work has shown that there are many other ways to address common malicious vectors in traditional networking and these malicious vectors also affect these new NFV environments such as the DDoS/DoS attack that was selected to develop this experimental work in a real NFV environment.

This work presented two strategies to address the selected attack and to prevent situations where the selected attack is not the cause of the service availability failure or a degradation of its performance.

Regarding to the detection and mitigation strategy that covers the selected DDoS/DoS attack, this work proves that the novel ML technique (GMM) implemented to prevent the attack was very powerful blocking around 1.3 million of DDoS/DoS packets (this amount of traffic represents around 90% of the incoming traffic in this test) sent by the attacker, allowing the Web server to continue to provide the service without any interruption.

On the other hand, there is another case when the service could be affected due to a huge volume of the legitimate requests sent through the web server. In this case, this work deploys a load balancing strategy that evidenced the benefit of using it, and it reduces the Web server CPU consumption.

Also the load balancing strategy decreases the amount of request sent to the principal web server, distributes them through the pool of web servers available in our NS, an additionally, it showed an important reduction of the web server latency resulting in an improvement of around 36% less time to reply to the received requests.

Finally, it was noted that the Web server never stops to replay requests and maintaining a high service availability.

To summarize, this work is an important step to continue working on improving the reaction against cyberattacks in a NFV environment. Also, this work shows our proposed GMM-based detection technique could be used on network security, opening the opportunities to try and test other techniques that had demonstrated their effec-

tiveness in other engineering areas. Moreover, providing a real NFV environment to try and test different works such as optimization models, ML models, among others, in our research group.

7.1 Publication Results

This research work allowed a conference paper published in the AnNet workshop of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM). The paper was called **"Detection of DDoS/DoS attacks: the UBM and GMM approach"** and was presented in a virtual session on May 17th - 2021.

Bibliography

[1] M. Chiosi, S. Wright Bell Canada, J. Erfanian, B. B. Smith, B. Briscoe, A. Reid, P. Willis CableLabs, D. Clarke, C. Donley CenturyLink, M. Bugenhagen, J. Feger, J. Benitez, N. Fischbach, K. Martiny, U. Michel DOCOMO, T. Nakamura, J. Triay Marques KDDI, K. Ogaki, T. Matsuzaki KPN, S. Zhang, A. K. de Boer, K. Ok, E. Kyoung PAIK, M. Stura, J. Carapinha, A. S. Gamelas Telecom, D. Lee, J. Han Park Softbank, R. Wakikawa, K. Nishi, S. Matsushima, M. Brunner, E. Demaria, A. Pinnola Telenor, P. Waldemar, G. Millstein, D. López, F. Javier Ramón Salguero, D. Kirkham, M. Ergen, M. Ahmet Karaman, A. Ulas, E. Lokman, N. Khan, R. Morera Vodafone, S. Sabater, A. Neal Windstream, and A. Nichols, "Network Functions Virtualisation (NFV) Network Operator Perspectives on Industry Progress Turk Telekom Argela," http://portal.etsi.org/NFV/NFV_White_Paper3.pdf, Tech. Rep., 2014.

[2] M. Pattaranantakul, R. He, Q. Song, Z. Zhang, and A. Meddahi, "Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures," *IEEE Communications Surveys and Tutorials*, 2018.

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[4] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, no. 4, pp. 34–41, 2005.

[5] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[6] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, sep 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7534741/

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

[8] M. Chiosi, D. Clarke, P. Willis, A. Reid CenturyLink, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, J. Benitez, U. Michel, H. Damker KDDI, K. Ogaki, T. Matsuzaki NTT, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. López, F. Javier Ramón Salguero, F. Ruhl, and P. Sen, "Network Functions Virtualisation," http://portal.etsi.org/NFV/NFV_White_Paper.pdf, Tech. Rep., 2012.

[9] M. Chiosi, S. B. Wright, D. Clarke, P. Willis CableLabs, C. Donley, L. Johnson CenturyLink, M. Bugenhagen, J. Feger, W. Khan, C. Cui, H. Deng, and C. Chen, "Network Functions Virtualisation (NFV) Network Operator Perspectives on Industry Progress," http://portal.etsi.org/NFV/NFV_White_Paper2.pdf, Tech. Rep., 2013.

[10] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973–993, 2014.

[11] S. Lal, T. Taleb, and A. Dutta, "NFV: Security Threats and Best Practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, may 2017.

[12] W. Yang and C. Fung, "A survey on security in network functions virtualization," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, jun 2016, pp. 15–19. [Online]. Available: http://ieeexplore.ieee.org/document/7502434/

[13] M. Pattaranantakul, R. He, A. Meddahi, and Z. Zhang, "SecMANO: Towards Network Functions Virtualization (NFV) Based Security MANagement and Orchestration," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, aug 2016, pp. 598–605. [Online]. Available: http://ieeexplore.ieee.org/document/7846998/

[14] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[15] T. Arias-Vergara, J. C. Vásquez-Correa, J. R. Orozco-Arroyave, and E. Nöth, "Speaker models for monitoring parkinson's disease progression considering different communication channels and acoustic conditions," *Speech Communication*, vol. 101, pp. 11–25, 2018.

[16] J. S. M. Osorio, J. A. V. Tejada, and J. F. B. Vega, "Detection of dos/ddos attacks: the ubm and gmm approach," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 866–871.

[17] M. Veeraraghavan, T. Sato, M. Buchanan, R. Rahimi, S. Okamoto, and N. Yamanaka, "Network function virtualization: A survey," *IEICE Transactions on Communications*, p. 2016NNI0001, 2017.

[18] J. d. J. Gil Herrera and J. F. Botero Vega, "Network Functions Virtualization: A Survey," *IEEE Latin America Transactions*, vol. 14, no. 2, pp. 983–997, feb 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7437249/

[19] M. Rahman, S. Iqbal, and J. Gao, "Load balancer as a service in cloud computing," in *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. IEEE, 2014, pp. 204–211.

[20] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," *ACM SIGCOMM computer communication review*, vol. 37, no. 4, pp. 1–12, 2007.

[21] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in *6th USENIX Symposium on Networked Systems Design and Implementation*, 2009, pp. 335–348. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558977.1559000

[22] W. D. Sincoskie, "A Survey Of Active Network Research - IEEE Communications Magazine," no. January, pp. 80–86, 1997.

[23] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, and A. Shaikh, "Design and Implementation of a Routing Control Platform," Tech. Rep., 2005.

[24] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A Clean Slate 4D Approach to Network Control and Management," Tech. Rep., 2005.

[25] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

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

[27] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[28] M. McCauley., "Pox," https://github.com/noxrepo/pox, 2012, last access: 06.04.2018.

[29] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[30] Floodlight, "Floodlight: A java-base openflow controller," https://floodlight.atlassian.net/wiki/spaces/HOME/overview?mode=global, 2012, last access: 06.04.2019.

[31] OpenDayLight, "Opendaylight: A linux foundation collaborative project," http://www.opendaylight.org, 2013, last access: 06.04.2019.

[32] N. Telegraph and T. Corporation., "Ryu," https://osrg.github.io/ryu/, 2008, last access: 06.04.2019.

[33] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[34] I. O. for Standardization ISO/IEC 27000, "Information technology — security techniques — information security management systems — overview and vocabulary." https://standards.iso.org/ittf/PubliclyAvailableStandards/c041933_ISO_IEC_27000_2009.zip, 2009, last access: 06.04.2019.

[35] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on scada systems," in *2011 International conference on internet of things and 4th international conference on cyber, physical and social computing*. IEEE, 2011, pp. 380–388.

[36] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[37] D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[38] C. Rasmussen, "The infinite gaussian mixture model," *Advances in neural information processing systems*, vol. 12, pp. 554–560, 1999.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[40] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[41] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE transactions on speech and audio processing*, vol. 2, no. 2, pp. 291–298, 1994.

[42] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[43] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[44] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 812–837, jan 2019.

[45] M. De Benedictis and A. Lioy, "On the establishment of trust in the cloud-based ETSI NFV framework," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2017*, vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., dec 2017, pp. 280–285.

[46] S. Ravidas, S. Lal, I. Oliver, and L. Hippelainen, "Incorporating trust in NFV: Addressing the challenges," in *Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks, ICIN 2017*. Institute of Electrical and Electronics Engineers Inc., apr 2017, pp. 87–91.

[47] M. Pattaranantakul, Y. Tseng, R. He, Z. Zhang, and A. Meddahi, "A first step towards security extension for NFV orchestrator," in *SDN-NFVSec 2017 - Proceedings of the ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, co-located with CODASPY 2017*. Association for Computing Machinery, Inc, mar 2017, pp. 25–30.

[48] C. Basile, A. Lioy, C. Pitscheider, F. Valenza, and M. Vallini, "A novel approach for integrating security policy enforcement with dynamic network virtualization," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–5.

[49] D. Montero, M. Yannuzzi, A. Shaw, L. Jacquin, A. Pastor, R. Serral-Gracia, A. Lioy, F. Risso, C. Basile, R. Sassu *et al.*, "Virtualized security at the network edge: A user-centric approach," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 176–186, 2015.

[50] "Datasheet: Vmware vcloud nfv," https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/solutions/vmware-vcloud-nfv-datasheet.pdf, accessed: 2021-08-24.

[51] L. S. Sampaio, P. H. Faustini, A. S. Silva, L. Z. Granville, and A. Schaeffer-Filho, "Using nfv and reinforcement learning for anomalies detection and mitigation

in sdn," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 432–00 437.

[52] G. Draper-Gil., A. H. Lashkari., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP,*, INSTICC. SciTePress, 2016, pp. 407–414.

[53] A. Habibi Lashkari., G. Draper Gil., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP,*, INSTICC. SciTePress, 2017, pp. 253–262.

[54] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[55] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network intrusion detection and prevention: concepts and techniques*. Springer Science & Business Media, 2009, vol. 47.

[56] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting http-based application layer dos attacks on web servers in the presence of sampling," *Computer Networks*, vol. 121, pp. 25–36, 2017.

[57] S.-N. Nguyen, V.-Q. Nguyen, J. Choi, and K. Kim, "Design and implementation of intrusion detection system using convolutional neural network for dos detection," in *Proceedings of the 2nd international conference on machine learning and soft computing*, 2018, pp. 34–38.

[58] F. S. d. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: an online approach for dos/ddos attack detection using machine learning," *Security and Communication Networks*, vol. 2019, 2019.

[59] "Cicflowmeter," https://www.unb.ca/cic/research/applications.html, accessed: 2021-01-04.

[60] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSP*, 2017, pp. 253–262.

[61] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[62] "Random forest classifiers- sklearn," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, accessed: 2021-01-04.

[63] T. Zhang, H. Qiu, L. Linguaglossa, W. Cerroni, and P. Giaccone, "Nfv platforms: Taxonomy, design choices and future challenges," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 30–48, 2020.

[64] A. Császár, W. John, M. Kind, C. Meirosu, G. Pongrácz, D. Staessens, A. Takács, and F.-J. Westphal, "Unifying cloud and carrier network: Eu fp7 project unify," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE, 2013, pp. 452–457.

[65] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 98–106, 2015.

[66] "Cloudband: Adopt lean operations and increase business agility," https://www.nokia.com/networks/solutions/cloudband/, accessed: 2021-08-24.

[67] R. Cziva and D. P. Pezaros, "Container network functions: Bringing nfv to the network edge," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 24–31, 2017.

[68] L. Li, K. Ota, and M. Dong, "Deepnfv: A lightweight framework for intelligent edge network functions virtualization," *IEEE Network*, vol. 33, no. 1, pp. 136–141, 2018.

[69] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene, "An open framework to enable netfate (network functions at the edge)," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–6.

[70] S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris, "Sonata: Service programming and orchestration for virtualized software networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017, pp. 973–978.

[71] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat, "xomb: Extensible open middleboxes with commodity servers," in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, 2012, pp. 49–60.

[72] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 323–336.

[73] M. Bezahaf, A. Alim, and L. Mathy, "Flowos: A flow-based platform for middleboxes," in *Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization*, 2013, pp. 19–24.

[74] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "Netbricks: Taking the v out of {NFV}," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 203–216.

[75] R. Riggio, I. G. B. Yahia, S. Latré, and T. Rasheed, "Scylla: A language for virtual network functions orchestration in enterprise wlans," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 401–409.

[76] P. Naik, A. Kanase, T. Patel, and M. Vutukuru, "libvnf: Building virtual network functions made easy," in *Proceedings of the ACM Symposium on Cloud Computing*, 2018, pp. 212–224.

[77] A. Alim, R. G. Clegg, L. Mai, L. Rupprecht, E. Seckler, P. Costa, P. Pietzuch, A. L. Wolf, N. Sultana, J. Crowcroft *et al.*, "{FLICK}: Developing and running application-specific network services," in *2016 {USENIX} Annual Technical Conference ({USENIX}{ATC} 16)*, 2016, pp. 1–14.

[78] M. Gallo and R. Laufer, "Clicknf: a modular stack for custom network functions," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 745–757.

[79] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "Nfv orchestration framework addressing sfc challenges," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 16–23, 2017.

[80] L. Diego, "Openmano: The dataplane ready open source nfv mano stack," 2015.

[81] "Open baton: An extensible and customizable nfv mano-compliant framework," https://openbaton.github.io/, accessed: 2021-08-24.

[82] W. Shen, M. Yoshida, K. Minato, and W. Imajuku, "vconductor: An enabler for achieving virtual network integration as a service," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 116–124, 2015.

[83] G. Xilouris, M.-A. Kourtis, M. J. McGrath, V. Riccobene, G. Petralia, E. Markakis, E. Palis, A. Georgios, G. Gardikis, J. F. Riera *et al.*, "T-nova: Network functions as-a-service over virtualised infrastructures," in *2015 IEEE Conference on Network*

Function Virtualization and Software Defined Network (NFV-SDN). IEEE, 2015, pp. 13–14.

[84] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, 2013, pp. 227–240.

[85] Y. Wang, G. Xie, Z. Li, P. He, and K. Salamatian, "Transparent flow migration for nfv," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 2016, pp. 1–10.

[86] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 163–174, 2014.

[87] B. Kothandaraman, M. Du, and P. Sköldström, "Centrally controlled distributed vnf state management," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015, pp. 37–42.

[88] M. Zhang, J. Bai, G. Li, Z. Meng, H. Li, H. Hu, and M. Xu, "When nfv meets ann: Rethinking elastic scaling for ann-based nfs," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–6.

[89] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, "Nfvnice: Dynamic backpressure and scheduling for nfv service chains," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 639–652, 2020.

[90] L. Zhang, C. Li, P. Wang, Y. Liu, Y. Hu, Q. Chen, and M. Guo, "Characterizing and orchestrating nfv-ready servers for efficient edge data processing," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[91] A. Tootoonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, and S. Shenker, "Resq: Enabling slos in network function virtualization," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 283–297.

[92] Y. Hu, M. Song, and T. Li, "Towards" full containerization" in containerized network function virtualization," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 467–481.

[93] R. V. Rosa, C. Bertoldo, and C. E. Rothenberg, "Take your vnf to the gym: A testing framework for automated nfv performance benchmarking," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 110–117, 2017.

[94] F. Moradi, C. Flinta, A. Johnsson, and C. Meirosu, "Conmon: An automated container based network performance monitoring system," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 54–62.

[95] F. Rath, J. Krude, J. Rüth, D. Schemmel, O. Hohlfeld, J. Á. Bitsch, and K. Wehrle, "Symperf: Predicting network function performance," in *Proceedings of the SIGCOMM Posters and Demos*, 2017, pp. 34–36.

[96] S. Geissler, S. Lange, F. Wamser, T. Zinner, and T. Hoßfeld, "Komon—kernel-based online monitoring of vnf packet processing times," in *2019 International Conference on Networked Systems (NetSys)*. IEEE, 2019, pp. 1–8.

[97] M. T. Raza, S. Lu, and M. Gerla, "vepc-sec: Securing lte network functions virtualization on public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3287–3297, 2019.

[98] H. J. Asghar, L. Melis, C. Soldani, E. De Cristofaro, M. A. Kaafar, and L. Mathy, "Splitbox: Toward efficient private network function virtualization," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, 2016, pp. 7–13.

[99] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 255–273.

[100] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. Duarte, "Bsec-nfvo: A blockchain-based security for network function virtualization orchestration," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[101] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-nfv: Securing nfv states by using sgx," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2016, pp. 45–48.

[102] A. Dhakal and K. Ramakrishnan, "Netml: An nfv platform with efficient support for machine learning applications," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 396–404.

[103] "Osm user guide," https://osm.etsi.org/docs/user-guide, accessed: 2021-08-24.

[104] "Openstack," https://www.openstack.org/, accessed: 2021-08-24.

[105] “Slowloris http dos,” https://github.com/XCHADXFAQ77X/SLOWLORIS, accessed: 2021-08-24.

[106] “Slowhttptest,” https://tools.kali.org/stress-testing/slowhttptest, accessed: 2021-08-24.

[107] “Hulk dos tool,” https://github.com/grafov/hulk, accessed: 2021-08-24.

[108] “Async hulk - https unbearable load king - hulk v3,” https://github.com/Hyperclaw79/HULK-v3, accessed: 2021-08-24.

[109] “Goldeneye,” https://github.com/jseidl/GoldenEye, accessed: 2021-08-24.

[110] “Goldeneye http denial of service tool,” https://packetstormsecurity.com/files/120966/GoldenEye-HTTP-Denial-Of-Service-Tool.html, accessed: 2021-08-24.

[111] “Apache jmeter,” https://jmeter.apache.org/, accessed: 2021-08-24.

[112] “Ddos/dos detection and mitigation strategies in a real a nfv environment,” https://github.com/jorgestivenm/GMM_OpenStack_implementation_real_time, accessed: 2021-08-24.