applied sciences

MDPI

Article

Detection and Mitigation of Security Threats Using Virtualized Network Functions in Software-Defined Networks

Manuel Domínguez-Dorado [1], Jesús Calle-Cancho [2,*], Jesús Galeano-Brajones [2], Francisco-Javier Rodríguez-Pérez [2] and David Cortés-Polo [2]

[1] Department of Domains, Systems and Digital Toolkit, Public Business Entity Red.es., 28020 Madrid, Spain; manuel.dominguez@red.es
[2] Department of Computing and Telematics Engineering, Universidad de Extremadura, Avd. Universidad S/N, 10003 Cáceres, Spain; jgaleanobra@unex.es (J.G.-B.); fjrodri@unex.es (F.-J.R.-P.); dcorpol@unex.es (D.C.-P.)
* Correspondence: jesuscalle@unex.es

Abstract: The evolution of interconnected systems and the evolving demands in service requirements have led to data centers integrating multiple heterogeneous technologies that must coexist. Consequently, the resource management and the security of the infrastructure are becoming more complex than in traditional scenarios. In this context, technologies such as Software-Defined Networking (SDN) or Network Function Virtualization (NFV) are being embraced as mechanisms that facilitate communication management. The integration of both technologies into a single framework, termed Software-Defined NFV (SDNFV) introduces a multitude of tools for managing the security of the data center's resources. This work delineates the primary characteristics of the evolution of these communication networks and their application to information security and communications within a data center. It presents an illustrative use case demonstrating the application of these next-generation technologies to detect and mitigate a security issue through virtualized network functions deployed in containers.

Keywords: NFV; SDN; security threats; detection; mitigation; dockers

Citation: Domínguez-Dorado, M.; Calle-Cancho, J.; Galeano-Brajones, J.; Rodríguez-Pérez, F.-J.; Cortés-Polo, D. Detection and Mitigation of Security Threats Using Virtualized Network Functions in Software-Defined Networks. *Appl. Sci.* **2024**, *14*, 374. https://doi.org/10.3390/app14010374

Academic Editor: Juan-Carlos Cano

Received: 16 December 2023
Revised: 29 December 2023
Accepted: 30 December 2023
Published: 31 December 2023

1. Introduction

The service provision landscape has undergone a paradigm shift in recent years, driven by technological advancements and evolving user demands. Legacy networks, designed for a simpler era, are ill-equipped to address the emerging networking challenges associated with this transformation. The ubiquitous adoption of cloud computing, server virtualization, BYOD (Bring Your Own Device) environments, and the exponential growth of big data and Artificial Intelligence (AI) exemplify the new paradigms necessitating advanced network capabilities [1].

Implementing any change in conventional networks, such as adding, relocating, or removing a network device, necessitates an overhaul of the entire network configuration. Networks comprise multiple devices (switches, routers, firewalls, load balancers, etc.), often from diverse vendors, resulting in intricate configurations that demand extensive time for implementation. The evolving needs of businesses demand enhanced networking capabilities, such as increased bandwidth, Quality of Service (QoS) control, and the convergence of voice, data, and video traffic. Legacy networks lack the dynamic agility necessary to effectively manage these demands [2].

Furthermore, scalability is a major hurdle for legacy networks. Networks typically expand to accommodate new users and their data demands, but this growth often outpaces the capacity of legacy architectures. This results in a network that is increasingly overloaded and unreliable. The need for a new abstraction layer is evident, one that enables the seamless addition of new devices while minimizing the burden on network administrators.

Another significant challenge is the inability to establish consistent policies across the network. Maintaining security or QoS policies within legacy networks is an arduous task. Every network alteration requires the manual configuration of numerous devices, making it prone to errors and inconsistencies. In BYOD environments, where employees use personal mobile devices, policy enforcement becomes even more challenging, increasing the vulnerability of the corporate network to security breaches. The surge in mobile devices and users, combined with the intricacy of traditional networking, has created a perfect storm that hinders the consistent application of policy compliance mechanisms.

Therefore, a fundamental shift in network architecture is necessary to overcome the limitations of legacy infrastructure. There is a need for a more adaptable, scalable, and secure network architecture that can seamlessly integrate with the evolving landscape of service provision [3].

Thus, Software-Defined Networking (SDN) [4] and Network Functions Virtualization (NFV) [5] architectures bring numerous benefits to this new computing paradigm. Their integration under a single framework, termed Software-Defined NFV (SDNFV) [6], is an active area of research and industry focus. This unified framework capitalizes on the control inherited from SDN and the flexibility gained from NFV's virtualized functions, allowing significant improvements in security issue detection and mitigation through SDNFV. Also, they are essential tools for managing security in this paradigm [7] because they are usually complementary technologies. In general, SDNFV-based networks combine SDN's network management with NFV's virtualization of network node functions, as well as the simplification of resource and service utilization within the network.

Due to the integration of diverse technologies and protocols in network orchestration, security assumes a pivotal role in enhancing communication security through the deployment of innovative mechanisms. Traditional security measures like Intrusion Detection Systems (IDS), firewalls, and others are traditionally stationed at the network periphery to fend off external threats. However, the advent of new network architectures necessitates novel security mechanisms to fortify services and networks, such as those employed in traditional deployments [8].

Network security has become a prominent theme for both industry and academia, with a surge of interest in its implementation within SDN [8] and NFV [9] architectures. Some of these implementations take advantage of the programmability of the network by applying solutions based on machine learning to detect anomaly flows [10] or implement a softwarized IDS in the SDN network [11] or develop NFV functions as firewalls to be deployed at the edge [12].

SDNFV technologies embody a novel and sophisticated paradigm to address security issues linked with new service deployments. Varied applications can be devised to manage security concerns of services or applications within the network using the benefits of SDNFV architecture.

In this work, a threat detection and mitigation algorithm is presented that takes the advantages provided by SNDFV-based networks. The algorithm employed leverages information entropy, a fundamental concept introduced by Shannon [13], to quantify uncertainty within network traffic patterns. Information entropy has demonstrated its efficacy in characterizing information content across a multitude of domains and applications like big data, fuzzy logic, or medicine [14–17]. To measure uncertainty within the network, the algorithm calculates the entropy of four key packet parameters: source IP address, destination IP address, source port, and destination port. These entropy values subsequently serve as the foundation for initiating appropriate mitigation techniques within the network infrastructure.

The remainder of this article is organized as follows: Section 2 introduces Network Function Virtualization, Software-Defined Networks, and their relationship. Section 3 details the SDNFV network architecture used in this work, along with its design and protocol development. Section 4 presents threat detection and mitigation in an SDNFV-based network, describing the algorithm for detecting potential security issues and the

mechanisms to mitigate them using virtualized network functions in a real scenario. Finally, Section 5 provides the concluding remarks of this study.

2. Enabling Technologies in Next-Generation Network Communications: SDN and NFV

SDN technology has emerged as a new network paradigm, characterized by separating the data plane from the control plane, aiming to simplify the management and configuration of traditional networks [18]. This shift in control, previously tightly integrated into individual network devices, allows the underlying infrastructure to be abstracted for applications and network services, treating the network as a logical or virtual entity.

Figure 1 illustrates the basic architecture of SDN from a logical perspective. As depicted, the SDN architecture is divided into three layers: application, control, and infrastructure.

- Applications Layer: These software programs execute specific tasks within an SDN environment, supplanting and expanding functions traditionally embedded in hardware devices of a conventional network. Examples of SDN applications include load balancing, security, or traffic engineering.
- Control Layer: Serving as the core intelligence of an SDN network, the SDN controller receives instructions and requirements from the application layer. It translates and relays these instructions to network devices in the infrastructure layer. Centralized network management through automated SDN applications facilitates the deployment and modification of network services, making it quicker and more straightforward.
- Infrastructure Layer: This layer is composed of the devices, which are network components that implement open standards (e.g., OpenFlow), enabling them to control forwarding and data processing capabilities within the network.

The control plane and data plane are distinct layers resulting from the separation of control and forwarding functions, providing applications with more network state information compared with protocols used in traditional networks. This enhanced information dissemination is facilitated by the presence of the network controller proposed by SDN.

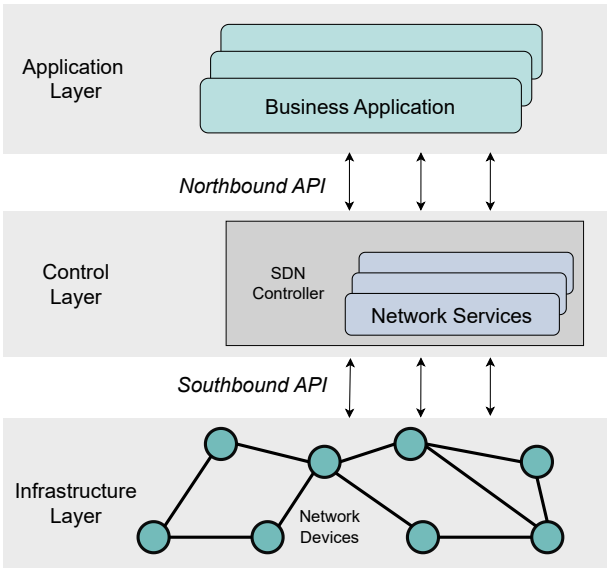


Figure 1. SDN functional architecture.

This technology provides agility, enabling dynamic flow management and optimizing resources for the changing needs of applications run by users in the cluster, which may have varying requirements in terms of features and QoS [19].

Communication and interaction between SDN components occur through APIs (Application Programming Interfaces). The API between defined SDN applications and the SDN controller is commonly referred to as the northbound API. Conversely, the API defined between the SDN controller and SDN network devices is known as the southbound API.

The SDN technology provides a series of important benefits that are outlined below:

- Improved automation and management.
- Deliver new network services quickly and easily.
- Implement a wide range of network policies.
- Reduced costs.
- Increased agility.
- Improved security.

On the other hand, NFV offers significant advantages in provisioning services in next-generation networks. This paradigm's primary goal is to decouple network functions from the physical devices on which they run.

Moreover, NFV has the potential to facilitate and enhance the deployment of new services with greater agility [20], allowing for meeting the low latency and high-reliability requirements needed by applications [21]. Typically, deploying new sophisticated network services like firewalling, load balancing, IPS (Intrusion Prevention System), IDS, routing, or WAN optimization demands the acquisition of specialized and costly hardware-based appliances by enterprises. This process also involves the installation of new equipment, which requires physical space, increases energy costs, demands specialized knowledge, and occasionally presents integration challenges. NFV seeks to revolutionize the architecture of network operators by leveraging standard virtualization technology. This transformation aims to consolidate various types of network equipment onto standard high-volume servers, switches, and storage solutions prevalent in data centers, network nodes, and end-user premises. NFV entails implementing network functions in software capable of operating on diverse industry-standard server hardware. These functions can be flexibly moved or instantiated at different locations within the network as needed, eliminating the necessity for new equipment installations.

The principal attributes of NFV encompass the following:

- Flexibility: NFV architecture facilitates the swift and straightforward deployment, installation, and provisioning of novel network services, thereby expediting Time-to-Market to meet the demands of businesses and users.
- Cost-effectiveness: NFV eliminates the requirement for costly hardware-based appliances by allowing the emulation of these devices via virtualization on standard high-volume servers, which are notably more economical.
- Scalability: NFV enables the deployment of new services or machines across multiple servers, obviating the necessity for additional physical space and simplifying network scalability to align with business requirements.
- Security: Network operators can administer and oversee the network while permitting their customers to securely manage their own virtual space and firewall within the network.
- Ubiquity: NFV facilitates the deployment of network services worldwide through virtualization, ensuring global availability.

Frequently, IT experts place SDN and NFV together because they share common objectives, as depicted in Figure 2. The primary aim of both SDN and NFV is to logically manage the network using software, reducing manual interaction with network devices. Hence, SDN and NFV paradigms are closely related [5], and with the efficient integration of both, significant cost savings and greater flexibility in service provisioning could be achieved. The integration of these paradigms into a single environment is known as an SDNFV network.

Finally, although both solutions are complementary, they are not mutually dependent and can be implemented separately, allowing for the deployment of SDN, NFV, or a combination of both. According to the NFV white paper [22], the objective of virtualizing network functions is to enable their use without applying SDN mechanisms, relying on the techniques currently prevalent in most data centers. However, approaches based on the separation of the control and forwarding planes, as proposed by SDN, improve

performance, simplify compatibility with existing deployments, and facilitate management and maintenance protocols.

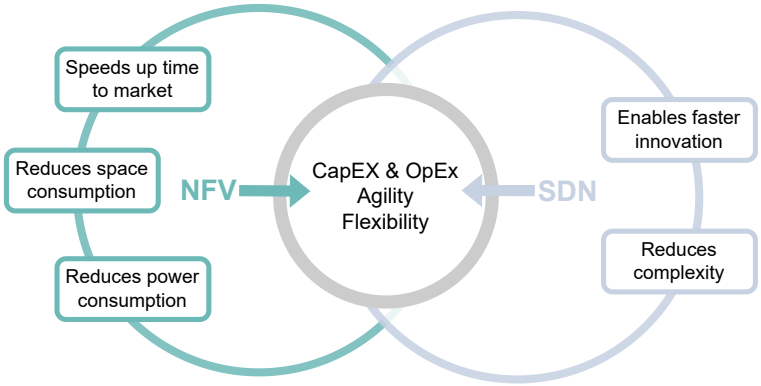


Figure 2. Relationship between SDN and NFV.

3. Security in SDNFV

Information security has perennially been a critical concern in the digital era due to the paramount value of information and the imperative need for its protection. Information security encompasses preventive and responsive measures taken by individuals, organizations, and technological systems to safeguard and preserve the confidentiality, integrity, and availability of information, often referred to as the CIA triad. This triad serves as a foundational model guiding information security policies within organizations, as outlined in ISO/IEC 27000:2018 [23]:

- Confidentiality: Ensures that information is not disclosed to unauthorized individuals, entities, or processes.
- Integrity: Ensures the accuracy and completeness of information.
- Availability: Ensures information is accessible and usable upon demand by authorized entities.

While the CIA triad represents fundamental properties in information security, additional properties like authenticity, accountability, nonrepudiation, and reliability can also be critical.

Networks play a key role in information systems, facilitating the exchange of information among various computers and resource distribution. A secure information system necessitates a secure network. The advent of novel network architectures has accommodated new business requisites; however, it has concurrently introduced fresh threats and vulnerabilities to the CIA triad that necessitate attention. For this reason, it is important to analyze vulnerabilities and threats associated with SDN and NFV architectures.

3.1. SDN Security Issues

As discussed in the previous section, the primary objective of SDN architecture is the separation of data and control planes. Consequently, the SDN architecture (refer to Figure 1) comprises three layers: the application layer, the control layer, and the infrastructure layer, each containing specific subcomponents:

- Network elements (NE) situated in the infrastructure layer.
- SDN controllers located in the control layer.
- SDN-enabled applications situated in the application layer.

While legacy networks possess a single type of component (network devices), the SDN architecture encompasses multiple elements (NE, SDN controllers, SDN applications, and northbound and southbound interfaces). Therefore, there is a need to protect not only network devices but also controllers, applications, and their communications.

Prior works [24] have analyzed each component of the SDN architecture using the Microsoft STRIDE methodology and identified potential threats to which they are vulnerable. Table 1 summarizes the potential threats.

Table 1. SDN Risks Analysis.

Attack Type	Security Property	SDN NE	SDN Controller	SDN App.
Spoofing	Authentication	Vulnerable	Vulnerable	Vulnerable
Tampering	Integrity	Vulnerable	Vulnerable	Vulnerable
Repudiation	Nonrepudiation	-	Vulnerable	-
Information Disclosure	Confidentiality	Vulnerable	Vulnerable	Vulnerable
Denial of Service (DoS)	Availability	Vulnerable	Vulnerable	Vulnerable
Elevation of Privileges	Authorization	Vulnerable	Vulnerable	-

The northbound interface (NBI) presents a significant attack vector due to the multiplicity of APIs employed by SDN controllers. These APIs leverage diverse technologies and languages, such as Python, Java, C, REST, XML, JSON, FTP, LDAP, and others. Exploiting vulnerabilities in any of these technologies or programming languages could grant an attacker control over the SDN network through the compromised controller. For instance, a compromised NBI could enable an attacker to create malicious SDN policies and manipulate the network environment.

Similarly, the southbound interface (SBI) constitutes a potential attack vector. Numerous APIs and protocols facilitate communication between the controller and network elements. These protocols include OpenFlow, Simple Network Management Protocol (SNMP), Secure Shell (SSH), NETCONF, OVSDB (Open vSwitch Database Management Protocol), OF-Config (OpenFlow Management and Configuration Protocol), etc. While each protocol utilizes its security methods, their relative novelty and potential implementation flaws leave them vulnerable. An attacker could exploit these vulnerabilities to modify or create malicious flows within a device's flow table, enabling the introduction of illegal traffic or the manipulation of routing for malicious purposes, such as Man-in-the-Middle (MITM) attacks.

The relative novelty of SDN and its software-based infrastructure presents a distinct challenge in terms of security. The lack of a historical record of security incidents hinders our ability to anticipate and proactively address potential attack vectors. This necessitates a proactive approach to network hardening, emphasizing robust security measures across all interfaces and protocols within the SDN architecture.

To improve the security of SDN networks, several measures can be implemented. A strategy is to secure the controller, which is considered the network's core. So, if this component is compromised, the overall functioning of the network is affected [25]. Another measure is to address the communication bottleneck between the controller and the switches, which can be exploited by attackers [26].

3.2. NFV Security Issues

In the actual networking landscape, the deployment of Virtual Network Functions (VNFs) must ensure that the robust security features inherent in legacy networks are preserved. While NFV offers numerous benefits, it also introduces new security concerns related to orchestrator and hypervisor protection, ubiquitous virtual appliances, third-party access, shared virtual machines and storage, and more.

VNFs, as network functions running on virtual machines, are susceptible to three categories of security threats:

- Generic virtualization threats: These include memory leakage, interrupt isolation, and other vulnerabilities inherent in virtualized environments.
- Threats specific to legacy network functions: These encompass existing threats previously targeted at physical network functions, such as flooding attacks and routing security vulnerabilities.
- New threats arising from the combination of virtualization and networking technologies: These threats are specific to NFV environments and exploit the unique characteristics of virtualized network functions.

The NFV security problem statement outlines these potential threats, including both novel threats and existing threats that manifest in new ways. To address these concerns, a security expert group designated by the European Telecommunications Standards Institute (ETSI) has provided comprehensive guidelines for securing NFV deployments [27]. These guidelines focus on the following key security areas:

- Topology validation and enforcement: Ensuring the network topology adheres to security policies and preventing unauthorized modifications.
- Availability of management support infrastructure: Guaranteeing the availability and integrity of critical infrastructure supporting NFV management.
- Secured boot: Implementing mechanisms that ensure only verified software is loaded during the boot process.
- Secure crash: Protecting system memory and state information in the event of a system crash.
- Performance isolation: Preventing resource starvation and ensuring fair resource allocation among VNFs.
- User/tenant authentication, authorization, and accounting (AAA): Implementing robust user and tenant authentication, authorization, and accounting mechanisms.
- Authenticated time service: Providing a reliable and tamper-proof time service for VNFs.
- Private keys within cloned Iimages: Protecting private keys used for encryption and authentication within cloned virtual machine images.
- Backdoors via virtualized test and monitoring functions: Preventing the introduction of unauthorized backdoors through virtualized testing and monitoring functions.
- Multiadministrator isolation: Ensuring the isolation of different administrative domains to prevent unauthorized access and privilege escalation.

By implementing these security measures and adhering to best practices outlined by ETSI, network operators can leverage the benefits of NFV while mitigating potential security risks and ensuring a secure and reliable network environment.

4. Implementation of Security Mechanisms into SDNFV-Based Networks

As previously discussed, while SDN and NFV offer significant benefits for modern network architectures, they also present security challenges for the network and its data. Integrating both approaches within a unified framework, known as an SDNFV-based network, offers improved mechanisms for detecting and mitigating security threats.

Due to the combined control and flexibility advantages inherited from SDN and NFV, respectively, SDNFV has become a hot topic in both research and industry. Several technologies have emerged to implement SDNFV frameworks due to the inherent complexity of its architecture and the relationship between the two approaches.

An SDNFV-based network defines complex network services that can operate on general-purpose hardware, replacing traditional dedicated hardware designed for specific functions [28]. This integration between NFV and SDN hardware is partially achieved through the decoupled control plane and data plane in SDN. The controller managing the network's control plane orchestrates the deployment of services, selecting the optimal location for each virtualized function.

While the controller's primary function in a pure SDN architecture remains managing the control plane and directing network flows, an SDNFV-based network requires additional functionality for orchestrating network resources, deploying virtualized functions, and managing their life cycle.

From a security perspective, the controller/orchestrator plays a vital role in handling key services for attack detection and mitigation, allocating network resources for black hole creation, and analyzing traffic flows, sources, and sinks.

Figure 3 illustrates the architecture and how both paradigms interconnect through the controller/orchestrator deployed in the network. As observed in the figure, the architecture is divided into different types of interconnected nodes. There are two types of SDN switches in the proposed SDNFV network architecture.

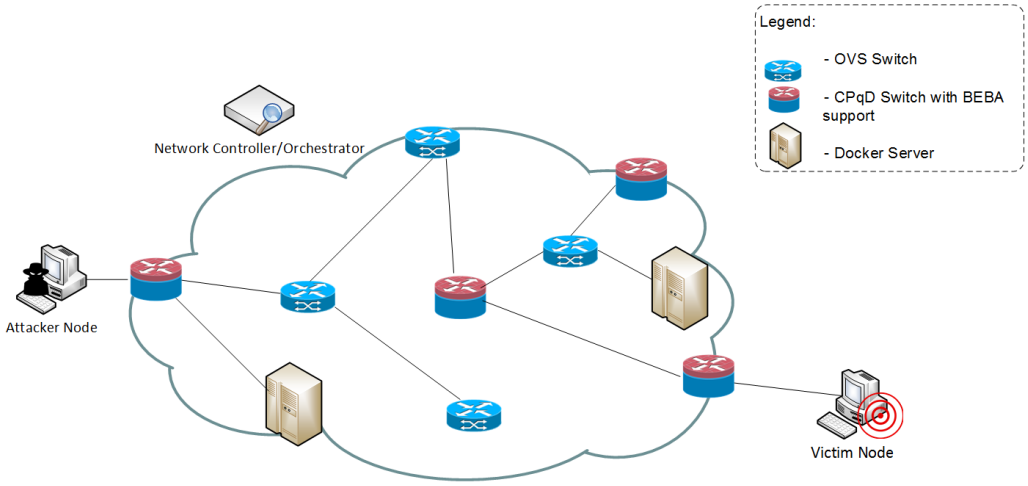


Figure 3. Proposed SDNFV network.

The first is built upon CPqD/ofsoftswitch13, which is an OpenFlow 1.3 Software Switch version executed in user space and implementing all standard functionalities. This switch has been modified by the BEhavioural-BAsed forwarding (BEBA) project workgroup to introduce an extended set of actions and primitives designed to monitor network traffic and enhance communication security[29]. The BEBA approach is an open-source implementation of an OpenState controller and switch, which is available at [30]. The second type of switch utilized within this setup is the Open vSwitch (OVS), functioning as a kernel module supporting OpenFlow, effectively replacing the Linux bridge implementation.

Another integral component is the RYU OpenFlow controller, managing the control plane of both switch types to facilitate intelligent routing within the SDNFV network. For the sake of simplicity, this controller also assumes the role of an orchestrator, overseeing the management of resources within the network function virtualization infrastructure (NFVI). These functions can be split into two entities in the network to separate the controller and orchestrator functionalities.

In this architecture, the infrastructure for deploying network functions virtualized utilizes a Docker infrastructure, operating within a Docker cluster. These Docker servers are integrated into the network as nodes, accessible through various routes. All virtualized functionalities are deployed within these servers, requiring the controller to adapt routes to incorporate NFV functionalities within the network and manage interactions with the network flows.

4.1. SDN Network

As highlighted in the preceding section, the SDN network incorporates two distinct switch types. OVS switches, functioning as nonintelligent switches, primarily operate by following packet switching rules established by the controller. They are also integrated into the Docker cluster to facilitate the deployment of Dockers that implement the NFV architecture.

The CPqD switches, designed with BEBA support, embody the proposal known as OpenState [31,32]. OpenState extends the core functionalities of OpenFlow, enabling the application of diverse match–action rules based on various states detected within the SDN flow tables of the switch.

This enhanced functionality empowers the switch to respond to packet-level events by analyzing the flows being switched. If the analysis aligns with the predefined rules in the flow tables, the switch can act following those rules.

OpenState operates as an extension of OpenFlow, integrating a traditional match/action flow table with an additional state table containing flow states. BEBA-enabled switches initially match packets with states from the state table before executing the flow table to determine the subsequent actions to take. Figure 4 illustrates the architecture of OpenState.

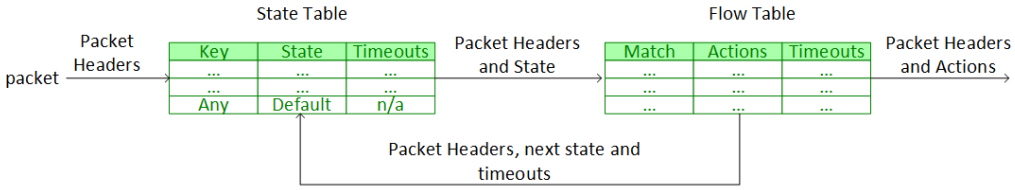


Figure 4. OpenState architecture.

4.2. NFV Architecture

The NFV architecture is implemented through the utilization of Dockers within a cluster. For effective orchestration and deployment of virtualized functionalities, the network controller necessitates comprehensive information about the cluster, encompassing hardware details and server IPs to establish the NFVI. With these data, the controller can efficiently allocate resources and deploy virtualized functionalities.

Within this ecosystem, network functions are encapsulated within lightweight Docker containers. This containerized approach ensures the independence of function deployment from the underlying platform. Figure 5 illustrates the NFVI architecture and the virtualization of functions within this framework.

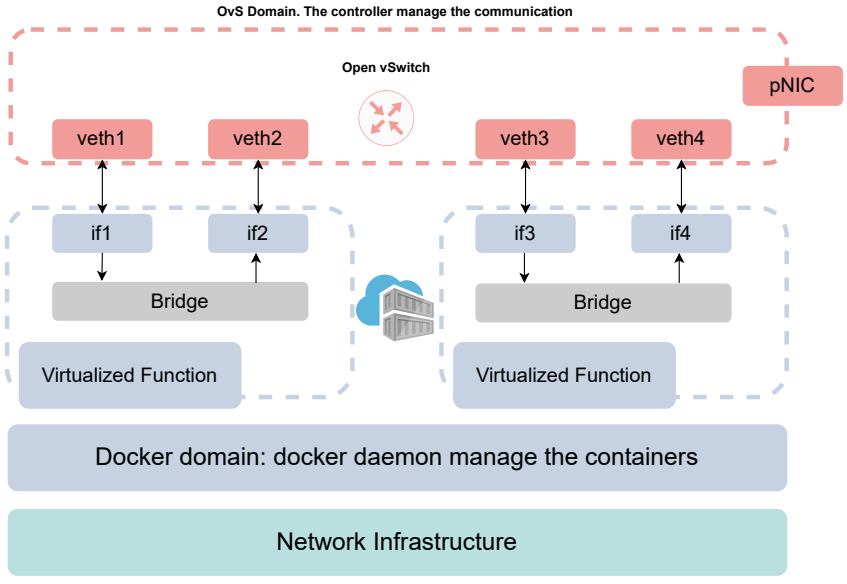


Figure 5. NFVI architecture.

The NFVI establishes connectivity with the SDNFV network via the physical network interfaces (pNIC). These interfaces facilitate packet exchange between the virtualized network functions and the SDN network. Each encapsulated virtual function within a container is equipped with two virtual network interfaces interconnected via a bridge (enabling dynamic virtual network configuration among containers using physical switch functions

for NFV infrastructure). This framework allows a small cluster to deploy numerous instances, configuring network connections and interfaces while facilitating communication between containers. Dockers implement container functionality through a file termed Dockerfile, containing instructions for automatic environment setup within a Docker image. This image, when instantiated within the NFVI, executes functions within the SDNFV network. Secure communication between the controller/orchestrator and the NFVI is ensured by encryption, facilitating the transmission of virtualized function configurations, the creation of Dockerfiles, and the deployment commands essential for NFV infrastructure.

4.3. SDNFV Controller/Orchestration

The architecture incorporates a RYU-based controller, which is a component-based SDN controller, featuring a set of predefined components essential for its functionality. These components are customizable within the controller application to adapt its behavior to specific problem requirements.

To support fundamental BEBA functionalities, various new messages, actions, and match fields must be integrated. The BEBA-supported controller extends the basic controller implementation, enabling the parsing of user-defined applications. It utilizes experimental messages to construct the application's logical model and a user-defined payload to convey essential information for traffic switching, flow decisions, or network rule execution.

The BEBA implementation has been adapted to incorporate the NFV orchestrator. It acquires information from the NFVI (in this scenario, the Docker cluster), deploys virtualized functions, and redirects traffic to the NFV. Figure 6 illustrates the comprehensive architecture of the controller/orchestrator.

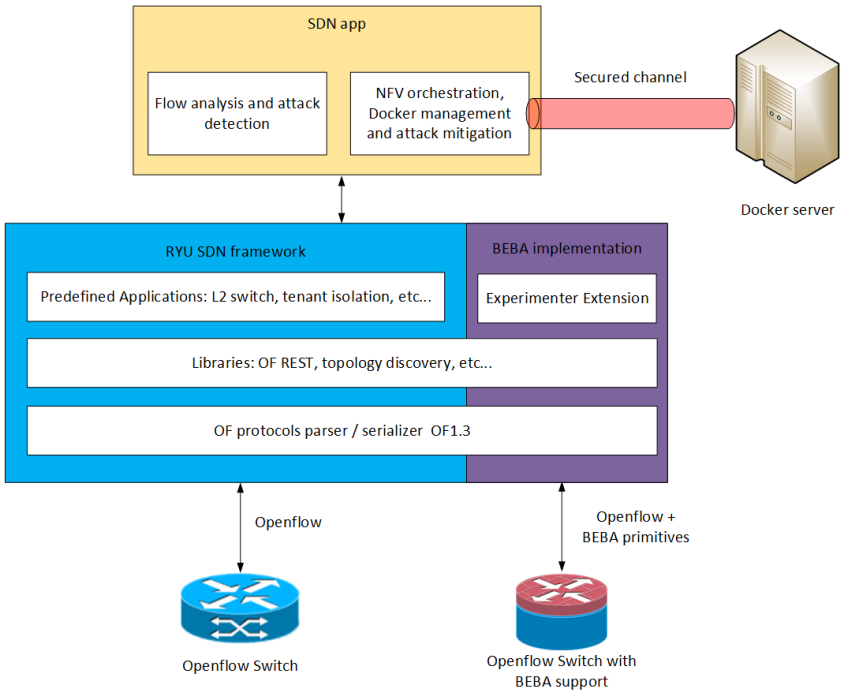


Figure 6. Architecture of the controller/orchestrator.

The RYU SDN framework is expanded through BEBA implementation utilizing the OpenFlow 1.3 protocol [33] and external libraries integrated into the original framework [34]. Notably, the SDNFV network is heterogeneous, demanding the controller to manage switches and implement a switching protocol for flow routing. OpenFlow switches with BEBA support execute normal applications (like predefined L2 switches).

The developed application built upon this framework leverages BEBA implementation to analyze flows and detect suspicious ones. Subsequently, the NFV orchestration module selects the optimal Docker server to deploy the container housing the NFV functionality.

It establishes a secure channel, executes deployment commands for the NFV, and upon successful deployment, reconfigures the flow tables to divert the identified flow to the NFV container, thus mitigating the attack.

5. Results

5.1. Detection and Mitigation

As described in the previous section, the controller/orchestrator integrates two modules designed for attack detection and mitigation. These modules are interlinked as the mitigation process necessitates detailed flow information, including source IP, source port, destination IP, destination port, and the attacking protocol. The detection module operates as a thread, routinely issuing requests to the OpenFlow switch to gather statistical information.

To obtain a response from the switch, an event handler was created to capture the messages containing the statistics' reply. These statistics undergo analysis to extract the packet's source and destination address, as well as the TCP or UDP source and destination port, to tally the different analyzed flows. Upon completing the flow analysis, entropy calculation ensues. Entropy is utilized for detecting Denial of Service (DoS) attacks by assessing statistical attributes within the packet header. Specifically, the analysis relies on comparing entropy across successive packet samples to identify an attack and can be defined as Equation (1):

$$E = -\sum_{i=1}^{n} p_i log_2 p_i \tag{1}$$

where E is the entropy, n is the number of elements detected in the flow analysis, and p_i is the probability of finding the *i-th* element in the conjunction of elements detected in the analysis. The application of entropy allows for the analysis of multiple variables within a flow by calculating the probability that the next packets encountered have similar information that the others processed previously. The attack performed in this experiment is a UDP flood attack by the malicious node, as a result of which the entropy of the flow sent by that node will be significantly reduced once packets of the same type start to be sent to the victim over the network. Once the entropy is calculated, the detection algorithm is executed. The algorithm is based on Equations (2) to (4).

$$lower - lim = \overline{x_p} - precision * \sigma_p \tag{2}$$

$$upper - lim = \overline{x_p} + precision * \sigma_p \tag{3}$$

$$Attack = \begin{cases} false & if \quad lower - lim \leq x \leq upper - lim \\ true & otherwise \end{cases} \tag{4}$$

where $\overline{x_p}$ is the mean value of the analyzed elements, and *precision* is the value used to define the precision in the detection algorithm. In this case, the values used for precision are 68% for low precision, 95% for medium precision, and 99.7% for high precision. Finally, σ_p is the standard deviation.

If an attack is detected, the controller/orchestrator searches a database containing information about the Docker cluster to select the server capable of efficiently implementing the NFV function. Once the server is chosen, the controller opens a secure channel to deploy the NFV in a Docker container. The deployment is carried out using the Dockerfile descriptor. This file describes the NFV function and the Docker's behavior. Algorithm 1 shows an example of a Dockerfile.

The Dockerfile defines the rules with which the Docker will be launched. Once the Docker container is built, the interfaces are created and the bridge between them is established. All of this is deployed on the chosen server from the Docker cluster. The controller, after completing the deployment phase, modifies the forwarding tables of the switches involved in the DoS attack communication to mitigate it. With this mechanism, the switches receiving the attack only need to forward the packets to the output port, and

the attack is mitigated using a server designed to absorb it without causing a DoS on the target node chosen by the attacker. In this case, the destination is the firewall deployed as an NFV function, a software implementation of a firewall with relevant rules to filter malicious traffic.

Algorithm 1 Example Dockerfile

```
# Firewall allowing traffic
# from port 80
FROM base
ENTRYPOINT ifinit && \
brinit && \
iptables -A FORWARD -p tcp && \
–dport 80 -j ACCEPT && \
iptables -A FORWARD -j DROP && \
/bin/bash
```

5.2. Testbed and Results

The scenario used is presented in Figure 7, and it is built on the top of Containernet and Docker servers infrastructure. As depicted in the figure, each infrastructure is deployed in a physical server interconnected by an Ethernet network.

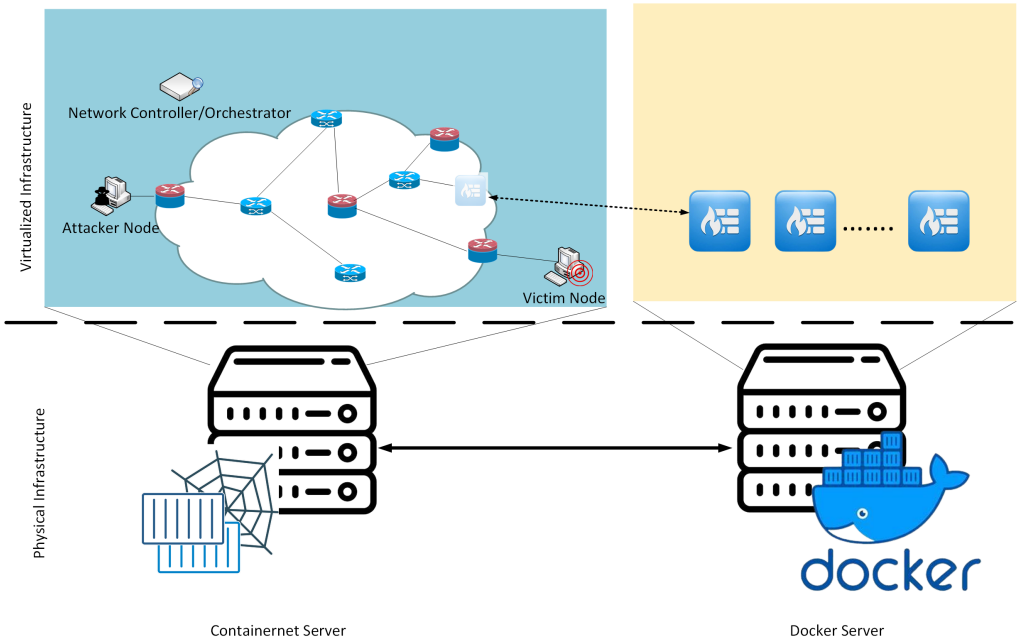


Figure 7. Scenario used to the experimental results.

The network was deployed in Containernet because it offers a fully virtualized environment based on Docker containers. These containers utilize sophisticated isolation features (e.g., Mount, UTS, IPC, PID, Network) to achieve a superior degree of sandboxing. This enhanced isolation capability is crucial to measure the resources used by the analyzed scenario.

Containernet is a fork of the widely recognized network emulator Mininet. While Mininet efficiently emulates specific use cases, its inherent limitations due to it not fully isolating emulated hosts have been a crucial issue for election.

In the scenario, a real trace with legitimate network traffic is inserted to the scenario to generate background traffic and make it difficult for the detection module of the algorithm. At around fifteen seconds, the attack is produced, and the attacker starts to generate illegitimate network traffic and send it to the victim. As can be observed in Figure 8a, the throughput is increased very quickly.

In this scenario, a real network traffic trace, obtained from project BEBA, is introduced to the environment to generate background traffic. This will complicate the algorithm's detection process and validate the process of attack mitigation. Around fifteen seconds after the beginning of the scenario, the attacker starts to generate illegitimate network traffic and send it to the victim. As depicted in Figure 8a, there is a rapid surge in throughput observed when the attack is started. In this scenario, the network does not implement any mechanism to mitigate the attack; hence, the throughput sent to the network is maintained during the experiment. As can be observed, the experiment was repeated 15 times to avoid random events in the scenario. Compared with the experiment with a mitigation mechanism based on the entropy analysis of the flows, the average throughput reached in the network is reduced by around 8%, and the attack is mitigated in less than 5 s, as can be observed in Figure 8b. At around the eighteenth second, the traffic starts to decrease in the network.

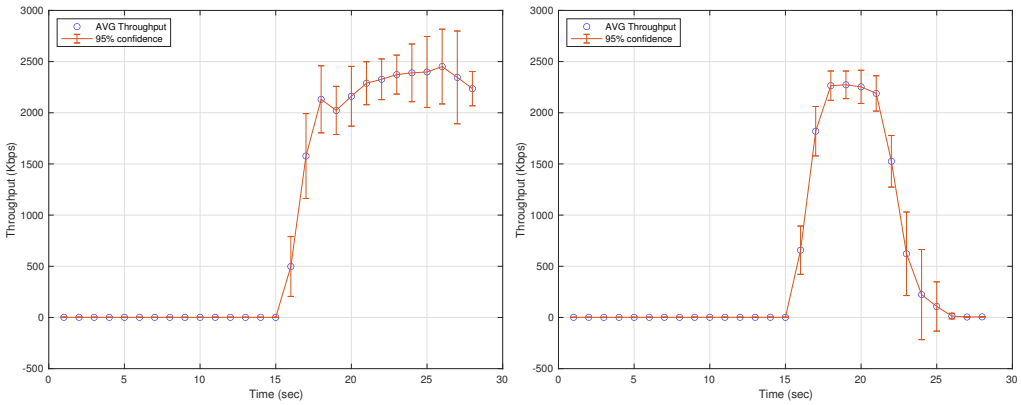


Figure 8. Network throughput: (**a**) scenario without any mitigation and (**b**) scenario with the proposed mitigation mechanism.

The entropy analysis, shown in Figure 9, studies the information stored in the controller and demonstrates the evolution of the entropy parameter in the source and destination IP address, which are the source and destination port of the attacking flow. As can be seen, when the attack is launched, the entropy level drops below the minimum, and the controller triggers the mitigation by redirecting the traffic over the network to the deployed Docker with the firewall service, which filters the packets belonging to the attack and leaves the rest of the packets that are sent and are legitimate.

Once the invalid traffic is filtered (around the twentieth second), the entropy returns to normal levels as the amount of packets sent to the victim has decreased.

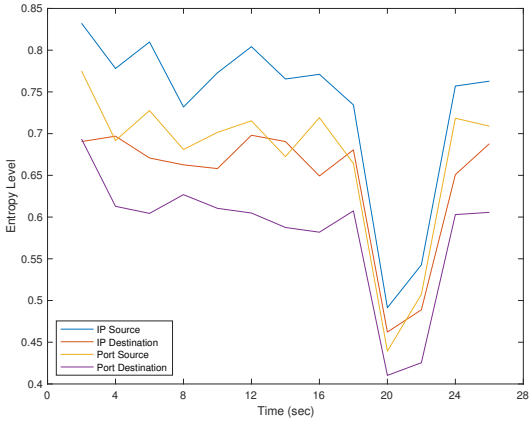
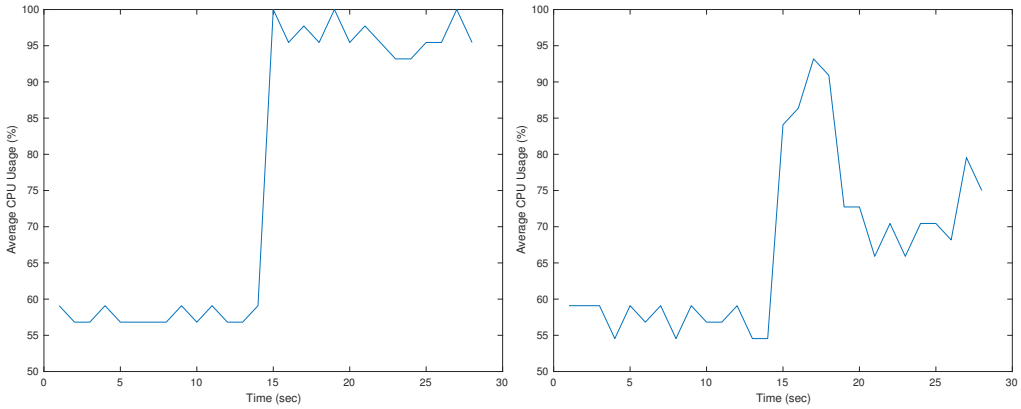


Figure 9. Entropy for each analyzed parameter.

Another of the most important parameters evaluated in this article is the average amount of CPU used in attacks, as shown in Figure 10a,b. The first shows the average CPU usage without mitigation, which is a continuous use throughout the experiment due to the need for the devices to receive the sent packets, process them, and forward them to the next hop, or attend to requests in case of a victim node. On the contrary, when the mitigation mechanism is active, it can be observed that it never reaches 100%, and the maximum CPU usage is achieved within seconds of the attack until it is detected. Additionally, it can be observed that the average CPU usage decreases once the network traffic is redirected to the virtualized NFV function.

As can be seen, the attacks analyzed in the scenario impact the network bandwidth as well as the CPU of the attacked devices. Thanks to the simplicity of the detection algorithm, it can be deployed on a controller with limited resources without impacting its normal operation. Additionally, as shown, this algorithm can detect and mitigate an attack in approximately 5 s, which is a short enough time not to saturate the CPU of the device, as shown in Figure 10.

This simple scenario allows for much greater development, where the implementation of new algorithms in the controller can analyze packets using much more complex and deep techniques, such as deep learning algorithms.



(**a**) (**b**)

Figure 10. CPU usage: (**a**) without any mitigation mechanism and (**b**) with the proposed mitigation mechanism.

6. Conclusions

This paper introduces a novel architecture that takes advantage of the benefits introduced by SDNFV-based architecture, and it allows the detection and mitigation of DoS attacks by utilizing virtualized functions within a Docker cluster. The presented architecture represents a straightforward and cost-efficient SDN-based network that scrutinizes flows and identifies attacks through an entropy mechanism developed within the controller. This mechanism demonstrates heightened capabilities in detection and mitigation by specifically targeting the flow involved in the attack. Moreover, it can discriminate among various attributes such as IP addresses, ports, or protocols.

The framework enables the deployment of diverse virtualized functionalities, which are contingent on the application running atop the RYU controller. While this work focuses on the presentation of the firewall virtualized function, the framework's versatility allows for the deployment of other functions, such as traffic conformation, packet deep inspection, sFlow collectors, and analyzers.

The potential of the framework lies in ensuring a transparent implementation of network functionalities for end-users. Moreover, the control plane is managed by the controller/orchestrator, efficiently handling network resources to enhance overall performance.

In future works, the algorithm used to detect and mitigate the DoS attack can be extended to detect other threats, as explained in Table 1. By leveraging the network's

programmability, the advantages offered by the OpenState framework, and the adaptability of virtualized functions, numerous other scenarios can be examined utilizing this innovative architecture. Additionally, the impact in more complex scenarios will be studied in depth in future work, where a greater number of nodes will be used to deploy the infrastructure to analyze the performance of the scenarios in detail.

Author Contributions: Conceptualization, M.D.-D., J.C.-C., and D.C.-P.; Methodology, J.C.-C., D.C.-P., and F.-J.R.-P.; Software, M.D.-D., J.C.-C., D.C.-P., J.G.-B., and F.-J.R.-P. Funding acquisition, D.C.-P.; Validation, M.D.-D., J.G.-B., and D.C.-P.; Formal analysis, M.D.-D., J.C.-C., and D.C.-P.; Investigation, M.D.-D., J.C.-C., and D.C.-P.; Resources, F.-J.R.-P. and D.C.-P.; Data Curation, J.C.-C.; Writing—original draft preparation, M.D.-D., J.C.-C., and D.C.-P.; Writing—review and editing, M.D.-D., J.C.-C., D.C.-P., J.G.-B., and F.-J.R.-P.; Supervision, D.C.-P. and J.C.-C.; Project Administration, D.C.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by TED2021-131699B-I00AEI/10.13039/501100011033/ Unión Europea NextGenerationEU/PRTR and by the Spanish Ministry of Science and Innovation [PID2020-112545RB-C54, PDC2022-133900-I00].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: Author Manuel Domínguez-Dorado was employed by the company Department of Domains, Systems and Digital Toolkit, Public Business Entity Red.es. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Salahdine, F.; Han, T.; Zhang, N. 5G, 6G, and Beyond: Recent advances and future challenges. *Ann. Telecommun.* **2023**, *78*, 525–549. [CrossRef]
2. Anerousis, N.; Chemouil, P.; Lazar, A.A.; Mihai, N.; Weinstein, S.B. The Origin and Evolution of Open Programmable Networks and SDN. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1956–1971. [CrossRef]
3. Munther, M.N.; Hashim, F.; Abdul Latiff, N.A.; Alezabi, K.A.; Liew, J.T. Scalable and secure SDN based ethernet architecture by suppressing broadcast traffic. *Egypt. Inform. J.* **2022**, *23*, 113–126. .: 10.1016/j.eij.2021.08.001 [CrossRef]
4. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
5. Mijumbi, R.; Serrat, J.; Gorricho, J.L.; Bouten, N.; De Turck, F.; Boutaba, R. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 236–262. [CrossRef]
6. Wood, T.; Ramakrishnan, K.K.; Hwang, J.; Liu, G.; Zhang, W. Toward a software-based network: Integrating software defined networking and network function virtualization. *IEEE Netw.* **2015**, *29*, 36–41. [CrossRef]
7. Martinez, H.F.; Mondragon, O.H.; Rubio, H.A.; Marquez, J. Computational and Communication Infrastructure Challenges for Resilient Cloud Services. *Computers* **2022**, *11*, 118. [CrossRef]
8. Correa Chica, J.C.; Imbachi, J.C.; Botero Vega, J.F. Security in SDN: A comprehensive survey. *J. Netw. Comput. Appl.* **2020**, *159*, 102595. .: 10.1016/j.jnca.2020.102595 [CrossRef]
9. Madi, T.; Alameddine, H.A.; Pourzandi, M.; Boukhtouta, A. NFV security survey in 5G networks: A three-dimensional threat taxonomy. *Comput. Netw.* **2021**, *197*, 108288. [CrossRef]
10. Ahmad, A.; Harjula, E.; Ylianttila, M.; Ahmad, I. Evaluation of machine learning techniques for security in SDN. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
11. Varghese, J.E.; Muniyal, B. An Efficient IDS Framework for DDoS Attacks in SDN Environment. *IEEE Access* **2021**, *9*, 69680–69699. [CrossRef]
12. Cziva, R.; Pezaros, D.P. Container Network Functions: Bringing NFV to the Network Edge. *IEEE Commun. Mag.* **2017**, *55*, 24–31. [CrossRef]
13. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
14. Jia, X.; Shang, L.; Zhou, B.; Yao, Y. An information entropy-based approach to outlier detection in rough sets. *Expert Syst. Appl.* **2010**, *37*, 6338–6344. [CrossRef]
15. Feng, G.; Li, Z.; Zhou, W.; Dong, S. Entropy-based outlier detection using Spark. *Clust. Comput.* **2020**, *23*, 409–419. [CrossRef]
16. Yuan, Z.; Chen, H.; Li, T.; Liu, J.; Wang, S. Fuzzy information entropy-based adaptive approach for hybrid feature outlier detection. *Fuzzy Sets Syst.* **2021**, *421*, 1–28. [CrossRef]

17. Combalia, M.; Codella, N.C.; Rotemberg, V.; Carrera, C.; Dusza, S.; Gutman, D. Validation of artificial intelligence prediction models for skin cancer diagnosis using dermoscopy images: The 2019 International Skin Imaging Collaboration Grand Challenge. *Lancet Digit. Health* **2022**, *4*, e659–e671. [CrossRef]
18. Nunes, B.A.A.; Mendonca, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634. [CrossRef]
19. Mahmoud, H.H.H.; Amer, A.A.; Ismail, T. 6G: A comprehensive survey on technologies, applications, challenges, and research problems. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4233. [CrossRef]
20. Adoga, H.U.; Pezaros, D.P. Network Function Virtualization and Service Function Chaining Frameworks: A Comprehensive Review of Requirements, Objectives, Implementations, and Open Research Challenges. *Future Internet* **2022**, *14*, 59. [CrossRef]
21. Banafaa, M.; Shayea, I.; Din, J.; Hadri Azmi, M.; Alashbi, A.; Ibrahim Daradkeh, Y.; Alhammadi, A. 6G Mobile Communication Technology: Requirements, Targets, Applications, Challenges, Advantages, and Opportunities. *Alex. Eng. J.* **2023**, *64*, 245–274. [CrossRef]
22. ETSI. Network Functions Virtualization, White Paper. 2012. Available online: https://www.etsi.org/technologies/nfv (accessed on 16 December 2023).
23. ISO/IEC. ISO/IEC 27000:2018. Technical Report ISO/IEC 27000:2018. 2018. Available online: https://www.iso.org/standard/73906.html (accessed on 16 December 2023).
24. ONF. Threat Analysis for the SDN Architecture. Technical Report TR-530. 2016. Available online: https://www.opennetworking.org/technical-communities/areas/services/1918-security (accessed on 16 December 2023).
25. Siddiqui, S.; Hameed, S.; Shah, S.A.; Ahmad, I.; Aneiba, A.; Draheim, D.; Dustdar, S. Toward Software-Defined Networking-Based IoT Frameworks: A Systematic Literature Review, Taxonomy, Open Challenges and Prospects. *IEEE Access* **2022**, *10*, 70850–70901. [CrossRef]
26. Shen, Y.; Wu, C.; Kong, D.; Cheng, Q. Flow Table Saturation Attack against Dynamic Timeout Mechanisms in SDN. *Appl. Sci.* **2023**, *13*, 7210. [CrossRef]
27. ETSI. NFV Security Requirements. Technical Report ETSI GR NFV-SEC 001. 2013. Available online: https://www.etsi.org/deliver/etsi_gs/nfv-sec/001_099/001/01.01.01_60/gs_nfv-sec001v010101p.pdf (accessed on 16 December 2023).
28. Zhang, T.; Qiu, H.; Linguaglossa, L.; Cerroni, W.; Giaccone, P. NFV Platforms: Taxonomy, Design Choices and Future Challenges. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 30–48. [CrossRef]
29. Bifulco, R.; Matsiuk, A. Towards Scalable SDN Switches: Enabling Faster Flow Table Entries Installation. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15), New York, NY, USA, 17–21 August 2015; pp. 343–344. [CrossRef]
30. BEBA. BEBA Project. Available online: https://github.com/beba-eu (accessed on 16 December 2023).
31. Bianchi, G.; Bonola, M.; Capone, A.; Cascone, C. OpenState: Programming Platform-Independent Stateful Openflow Applications inside the Switch. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 44–51. [CrossRef]
32. Wazirali, R.; Ahmad, R.; Alhiyari, S. SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Appl. Sci.* **2021**, *11*, 6999. [CrossRef]
33. OpenFlow. OpenFlow 1.3 Switch. Available online: https://github.com/CPqD/ofsoftswitch13 (accessed on 29 December 2023).
34. Fernandes, E.L.; Rojas, E.; Alvarez-Horcajo, J.; Kis, Z.L.; Sanvito, D.; Bonelli, N.; Cascone, C.; Rothenberg, C.E. The road to BOFUSS: The basic OpenFlow userspace software switch. *J. Netw. Comput. Appl.* **2020**, *165* , 102685. [CrossRef]