future internet

MDPI

Article

Misuse Patterns from the Threat of Modification of Non-Control Data in Network Function Virtualization

Abdulrahman K. Alnaim

Department of Management Information Systems, School of Business, King Faisal University, Al Ahsa 31982, Saudi Arabia; aalnaim@kfu.edu.sa

Abstract: Network Function Virtualization (NFV) is a virtual network model, the goal of which is a cost-efficient transition of the hardware infrastructure into a flexible and reliable software platform. However, this transition comes at the cost of more security threats. A key part of this virtualization environment is the hypervisor, which emulates the hardware resources to provide a runtime environment for virtual machines (VMs). The hypervisor is considered a major attack vector and must be secured to ensure network service continuity. The virtualization environment contains critical non-control data where compromise could lead to several misuses, including information leakage and privilege and resource modification. In this paper, we present a misuse pattern for an attack that exploits the security vulnerabilities of the hypervisor to compromise the integrity of non-control data in the NFV environment. Misuse patterns are used to describe how attacks are carried out from the attackers' perspective. The threat of modification of non-control data can lead to several misuses, and in this paper, we discuss three of them. The defenses to this attack can be incorporated into the Security Reference Architecture (SRA) of the NFV system to prevent these misuses.

Keywords: network function virtualization (NFV); cloud computing; virtualization; non-control data; hypervisor; misuse patterns

Citation: Alnaim, A.K. Misuse Patterns from the Threat of Modification of Non-Control Data in Network Function Virtualization. *Future Internet* **2022**, *14*, 201. https://doi.org/10.3390/fi14070201

Academic Editors: Vijayakumar Varadarajan, Nancy Victor and Buddhadeb Pradhan

Received: 19 June 2022
Accepted: 28 June 2022
Published: 30 June 2022

1. Introduction

Network service providers need to deploy network equipment such as firewalls, domain name servers (DNSs), load balancers, routers, and switches at the consumers' premises to deliver a network service. These network hardware devices may connect many computers with different operating systems and protocols, which increases the complexity of network infrastructure [1,2]. In addition, TSPs may require deploying additional network equipment to cover the consumers' needs [3], which expands the network infrastructure and increases the operational expenditure (OpEx) and the capital expenditures (CapEx), making managing the network infrastructure a cumbersome process [4]. The heterogeneity of these pieces of equipment also makes it difficult to have a secure network environment.

A different paradigm has emerged in the network industry that developed the network infrastructure and its service delivery. Network Function Virtualization (NFV) takes advantage of virtualization to deliver virtual network functions, i.e., virtual firewalls, virtual switches, etc. It promises independence in hardware and software development, because they are not integrated with each other, and reduces the OpEx, the CapEx, and even the total cost of ownership (TCO) [5,6]. NFV also ensures a sharable and scalable network environment in which many NFV consumers can share and scale the network resources provided by TSPs according to their requirements. We consider here TSPs as NFV providers.

Although NFV promises many benefits, as mentioned previously, it leads to security issues [7,8]. NFV providers are required to undertake substantial efforts to ensure a secure NFV service environment. To provide a secure NFV service, we need to study

and understand the possible threats. In [9], we looked at the main security threats in NFV and the possible countermeasures to these threats. In this survey, we classified the vulnerabilities and mapped them to their possible threats. Here, we use misuse patterns to describe one of these threats, the threat of maliciously modifying non-control data. Misuse patterns are used to describe how an attack is carried out from the point of view of the attacker [10]. They also define the environment in which the attack can be carried out, the possible countermeasures to mitigate it, and the forensic information that could be used to trace the attack once it happens [10]. The patterns are part of an ongoing catalog that can be used by system designers to consider security aspects when building an NFV system.

The threat of modifying non-control data has been studied in various systems. In [11], the authors demonstrated real-world applications vulnerable to such attacks to show how non-control data attacks are realistic. In [12], the authors described some possible cases of kernel non-control data attacks. In [13], a data-oriented programming technique was used to construct non-control data attacks on nine applications. The authors also used a dataflow stitching technique to generate data-oriented exploits that led to non-control data attacks. Another scenario of a non-control data attack is explained in [14], in which a memory corruption vulnerability was leveraged. Although many researchers have explained the possibility of this threat, no one in the literature has used patterns to analyze the threat of non-control data in an NFV system. Patterns have proved convenient to describe the threats in several environments, such as cloud [15], IoT [16], and VoIP [17].

The remainder of this paper is arranged as follows: Section 2 provides a background on Network Function Virtualization (NFV), architectural modeling using patterns, and the threat of modifying non-control data. Section 3 presents a misuse pattern that results in modifying a non-control data threat in NFV. Section 4 shows how this pattern could be used to extend the SRA from [18], while Section 5 considers related work. Section 6 contains conclusions and future work. Appendix A, at the end of the paper, describes a variation in the POSA template, which we use for misuse patterns [10]. We consider the POSA template to be more suitable for describing misuse patterns [19].

2. Background

2.1. Network Function Virtualization

NFV transforms the traditional network architecture from a static architecture that comprises physical hardware to an agile one that provides network functions as software running in virtual machines (VMs). Decoupling the network functions from its dedicated hardware and emulating them to virtual servers will result in the following benefits [19]:

- Flexibility: The network will be provided as a software service, ensuring flexible and faster deployment;
- Elasticity: NFV consumers will be able to dynamically scale the network resources;
- Extensibility: It would be possible to dynamically add more network services within the network service;
- Faster deployment: The network service will be configured faster.

The European Telecommunication Standards Institute (ETSI) introduced the first architecture of NFV, shown in Figure 1 [20]. It consists of three main components: the network function virtualization infrastructure (NFVI), virtualized network functions (VNFs), and NFV management and orchestration (MANO).

The NFV infrastructure is the foundation platform for the network service and contains hardware resources (storage, CPU, network, etc.); virtualized resources (virtual storage, virtual CPU, virtual network, etc.); and the virtualization layer, which contains the hypervisor. The hypervisor, also called the virtual machine manager (VMM), deploys VMs, emulates the necessary resources, and allows for resource sharing, while also ensuring isolation among them [21].

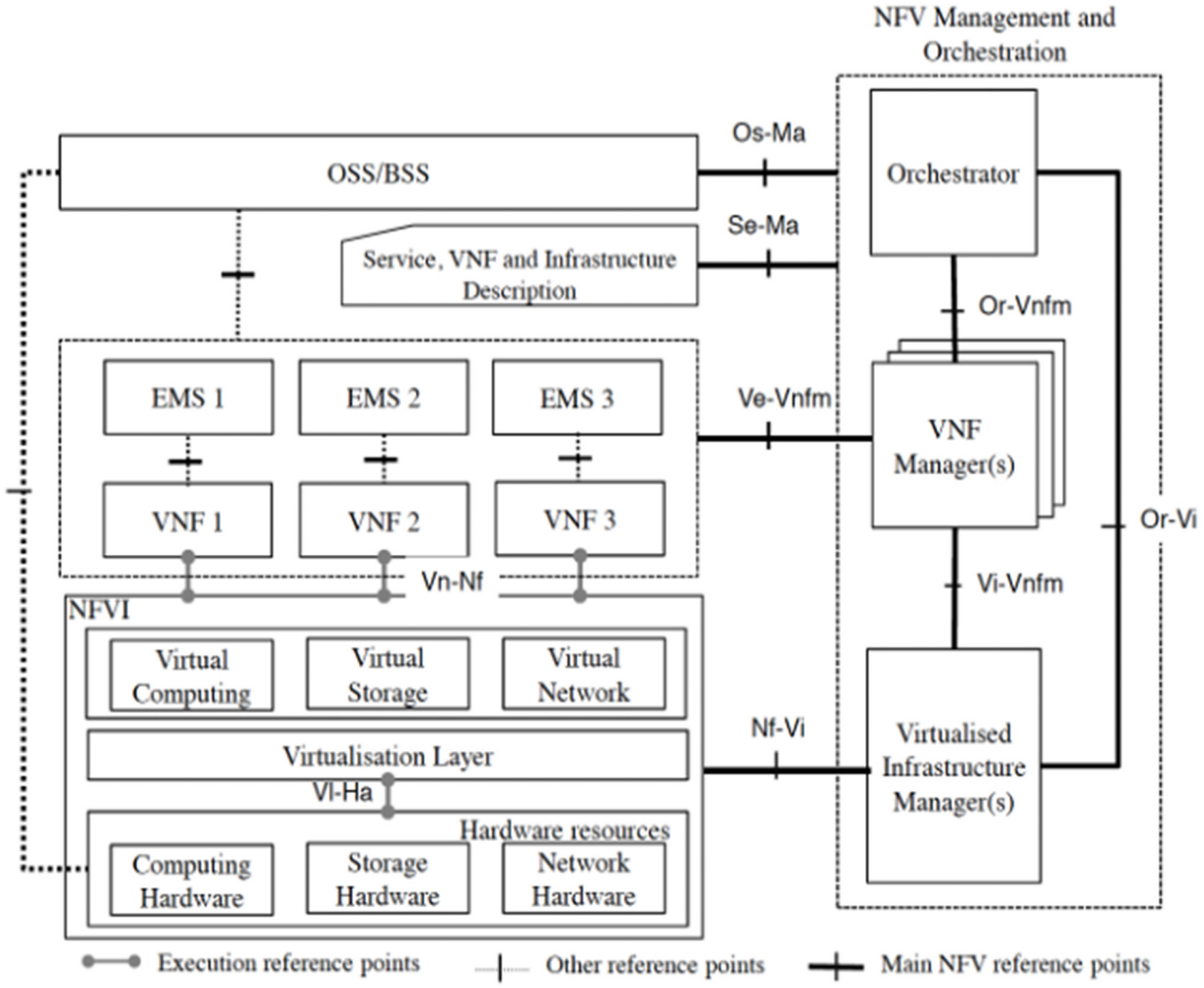

Figure 1. NFV Reference Architecture Framework [20].

Further, VNFs are software implementations of the network functions that are deployed on the NFVI. A single VNF may contain several components (VNFCs), which are software components of a VNF, or may contain only one network function in order to maintain its scalability. VNFs are hosted by VMs [22] or even a container [23].

The third component is the NFV MANO, which covers the lifecycle management and orchestration of the virtual network service. It contains three management units: the virtualized infrastructure manager (VIM), responsible for managing the interaction between the VNFs and the NFVI resources; the VNF manager, which manages and monitors the VNF resources; and the NFV orchestrator (NFVO), which provisions the necessary resources for the network service.

2.2. Patterns

A pattern is a solution to a recurrent problem in a given context. Patterns embody abstractions and provide common vocabularies for system designers. Their solutions are suggestions, not plug-ins [19], which means that they are prototypes and there are many ways to instantiate a pattern. There are several types of patterns, intended for specific design purposes. Design and architectural patterns are used to build the functional aspects of extendable systems [24]. Security patterns are used to build secure systems by defining a way of controlling vulnerabilities or stopping specific attacks [10]. Threat patterns describe the steps of an attack that could lead to several misuses [25]. Misuse patterns are used

to describe, from the attacker's perspective, a generic method of attacking a system by exploiting a vulnerability. They also describe the environment in which an attack may be performed, the possible countermeasures to mitigate it, and the method to find forensic information to trace the attack once it happens [10].

Patterns are described using templates; each template is different based on the type of pattern. For example, a misuse pattern template contains countermeasures, consequences, and forensic sections that are not available in a design pattern template. We use the Pattern-Oriented Software Architecture (POSA) template as we consider it more suitable for describing security aspects [10]. The descriptions of patterns may be written in textual language, and their solutions are usually shown in Unified Modeling Language (UML).

2.3. Modifying Non-Control Threat

Most security threats are related to altering the control flow of the targeted system, either by injecting a code or by reusing existing code such as return and call instructions [26,27]. It mainly refers to the data loaded in the processing counter during program execution, in which the attacker exploits, for instance, a memory corruption vulnerability, such as buffer overflow or integer overflow, to compromise the system [28,29]. It has been indicated that threats related to non-control data are also possible in real-world applications and are closely equivalent to control data threats [11,30]. Non-control data attacks do not affect the control flow of a system; instead, they are carried out by altering the non-control data of a targeted program, such as configuration data, decision-making data, user identity data, and user input [29]. There are also other critical non-control data in the kernel level susceptible to an attack, such as user privilege data, resource utilization data, and service policy data [12,31].

The threat of modifying non-control data is also possible in the virtualization environment. The hypervisor is being used as a virtualization layer in many systems, including Network Function Virtualization (NFV) [32], due to its ability to enable resource sharing and, at the same time, ensure isolation among virtual machines. Since hypervisors have a smaller code base, it has been assumed that they and the VMs running on top of them are secure [33–36]. However, some hypervisors indeed are quite complex and have large lines of codes. For instance, Xen contains more than 900K lines of codes [37]. Kernel-based Virtual Machine (KVM) contains around 850K lines of codes [38]. Continuing bug and exploit reports indicate that hypervisors are not secured, as has been assumed [39–42], and neither are non-control data, which are considered an exploitable attack vector [12,13,31]. In the following section, we show a misuse pattern from modifying non-control threats that could lead to several misuses in the NFV environment.

3. Misuse Patterns from the Threat of Modification of Non-Control Data in NFV

3.1. Intent

Threats related to resource modification are possible in the virtual environment. An attacker modifies the resource-utilization-related data (i.e., processing resources) of virtual machines with the aim to degrade their performance or even deny their services.

3.2. Context

Hypervisors emulate the necessary resources (i.e., processing, storage, and network) from the cloud infrastructure to virtual machines and provide isolation among them. The hypervisor contains critical data for each VM, which can be classified as control and non-control data. The non-control data include configuration data, user input data, etc.

3.3. Problem

How can an attacker compromise the integrity of non-control data of an NFV consumer's virtual machine, which may lead to several misuses? The attack can be carried out by exploiting the following vulnerabilities:

1. VMs communicate with their hosting hypervisor via hypercalls, which are low-level system calls for basic processing and resource access requests. Thus, it would be difficult to distinguish between malicious and legitimate hypercalls [39,43].
2. An attacker is able to take advantage of a malicious network service request translated by a VM to a malicious hypercall, which enables the attacker to access hypervisor files and modify its data [44–46];
3. In a virtualization environment, hypervisors are complex and have large lines of source codes, which increases the chances of vulnerabilities being present [40,47,48] and makes it difficult to maintain their security [49].
4. An attacker can exploit existing vulnerabilities to run arbitrary code, which leads to illegal access to host files [50–52].
5. Emerging attacks aim to execute existing codes within the host domain files instead of injecting new codes, such as return-oriented attacks [45]. Real-world software applications contain critical non-control data. Neglecting their protection may compromise their integrity, which will affect the virtual network service [11,53].
6. Because the network service is hosted in a sharable environment, if one VNF is compromised, it may affect the other VNFs that share the same environment [54].

3.4. Solution

We assume that the attacker is a regular NFV consumer who has a valid account with a network service provider. The attacker requests a network service that could include one or a bundle of VNFs. The hypervisor emulates resources from NFV infrastructure (NFVI) and assigns these resources to VMs. Processing and resource requests are sent as hypercalls. VMs pass these system calls to the hypervisor, which executes them in hardware. These hypercalls can be maliciously made for the purpose of controlling or accessing the hypervisor files [45,46]. There are several vulnerabilities that exploit a heap overflow and could help the attacker access the hypervisor files and arbitrarily modify the resource utilization data [50–52,55].

The hypervisor deals with emulated vCPUs and schedulers to ensure that VMs receive the necessary processing resources. Each VM is associated with two resource properties, called weight and cap. The weight indicates the amount of physical CPU resource that the VM receives, while the cap is an absolute value that represents the maximum share of resource each VM can consume. A cap of 100 means the VM can use one physical CPU, 50 means half a physical CPU, and 300 means three physical CPUs; the default value is 0, which indicates that there is no upper limit. These non-control data are critical and affect the network service when their integrity is compromised [11,53].

The non-control data are stored in a file within the hypervisor, called the domain structure [31]. Each VM has its own associated domain structure. In Xen hypervisors, for instance, the first domain is the privileged domain called Dom0, which creates other unprivileged VMs (Dom1, Dom2, DomN) [56]. The attacker can identify a victim's VM by looking at its domain_id, which is 0 for Dom0, 1 for Dom1, 2 for Dom2, etc. Therefore, by traversing Dom0, the attacker can identify users' domains [31].

Here is a possible scenario for modifying a non-control attack. The attacker, or the NFV consumer, runs a malicious application to construct a payload that calls the vulnerable hypercall in which the VM passes it to the hypervisor [13,57]. The result of the hypercall is access to the hypervisor files and implementation of a return-oriented attack (ROP) [58,59] that aims to modify codes already existing in files, instead of injecting new codes, to modify the weight and cap of a victim's VM. For example, if the VM has a weight of 256 and a cap of 4, the attacker may modify the weight to 128 and the cap to 1, which will reduce the amount of CPU the VM should receive. This attack is applied in the virtualization environment and possible in the NFV environment in a way to compromise the integrity of resource utilization of a network service [54,60].

3.5. Structure

Figure 2 shows a class diagram for modifying a non-control attack in NFV. The NFV consumer connects to the network service, or VNFs, through an application programming interface (API). VMs are created using virtual machine image (VMI); the image contains software configuration, including the operating system and application programs. VMs are created and managed by the hypervisor, which emulates the hardware resources from the NFV infrastructure (NFVI) and may contain vCPUs and schedulers. The VIM is a software unit that manages and monitors the NFVI. Lastly, the domain is a structure that stores the basic resource information of VMs, such as weight and cap.

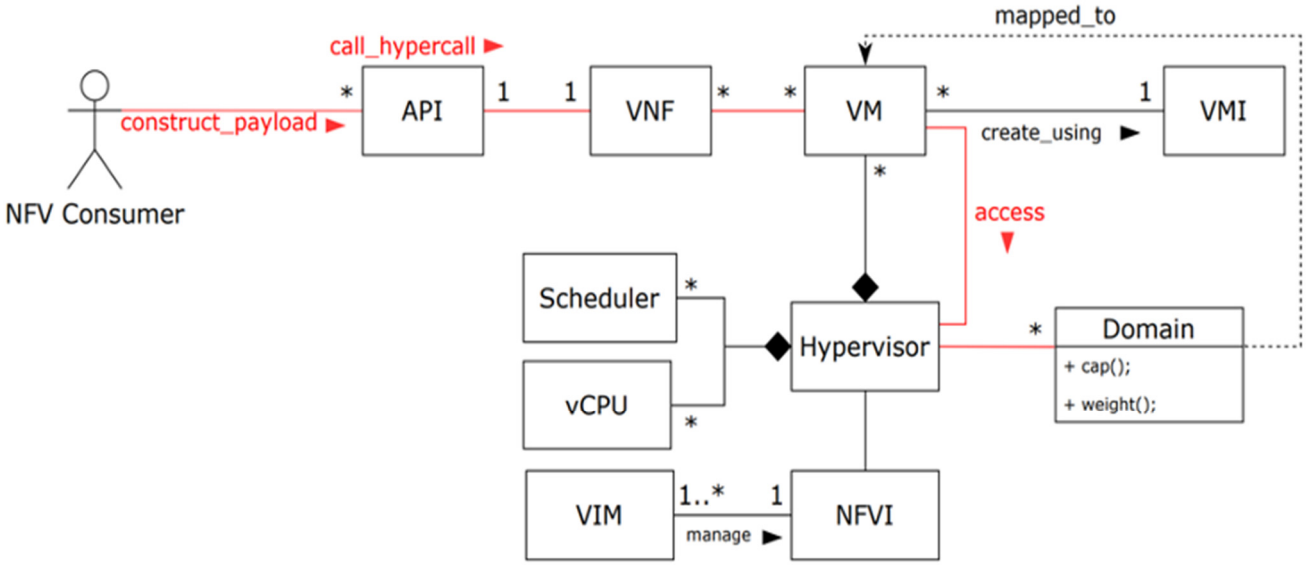

Figure 2. Class diagram for modifying a non-control attack in NFV.

3.6. Dynamics

Use case 1 (a misuse): Gain unauthorized access to hypervisor files by executing a malicious hypercall.

Summary: The attacker, who can be an NFV consumer, constructs a payload and writes a malicious application in the VNF that enables the attacker to make a malicious hypercall and access hypervisor files. The dynamics of this misuse are shown in Figure 3.

Actor: NFV consumer (attacker)

Precondition: The attacker has a valid account and active network services.

Description:

1. The attacker runs a malicious application.
2. The attacker constructs a payload that enables the attacker to make a hypercall.
3. The malicious hypercall is initiated and sent through the application.
4. The malicious hypercall is forwarded to the VNF.
5. The VNF passes the hypercall to its hosted VM.
6. The VM passes the hypercall to the hypervisor.
7. The hypervisor executes the malicious hypercall.
8. As a result of the hypercall execution, access is granted to the attacker.
9. The attacker is able to illegally access hypervisor files.

Postcondition: The attacker controls the hypervisor and has direct access to its files.

Use case 2 (a misuse): Degrade the performance of the victim's network service.

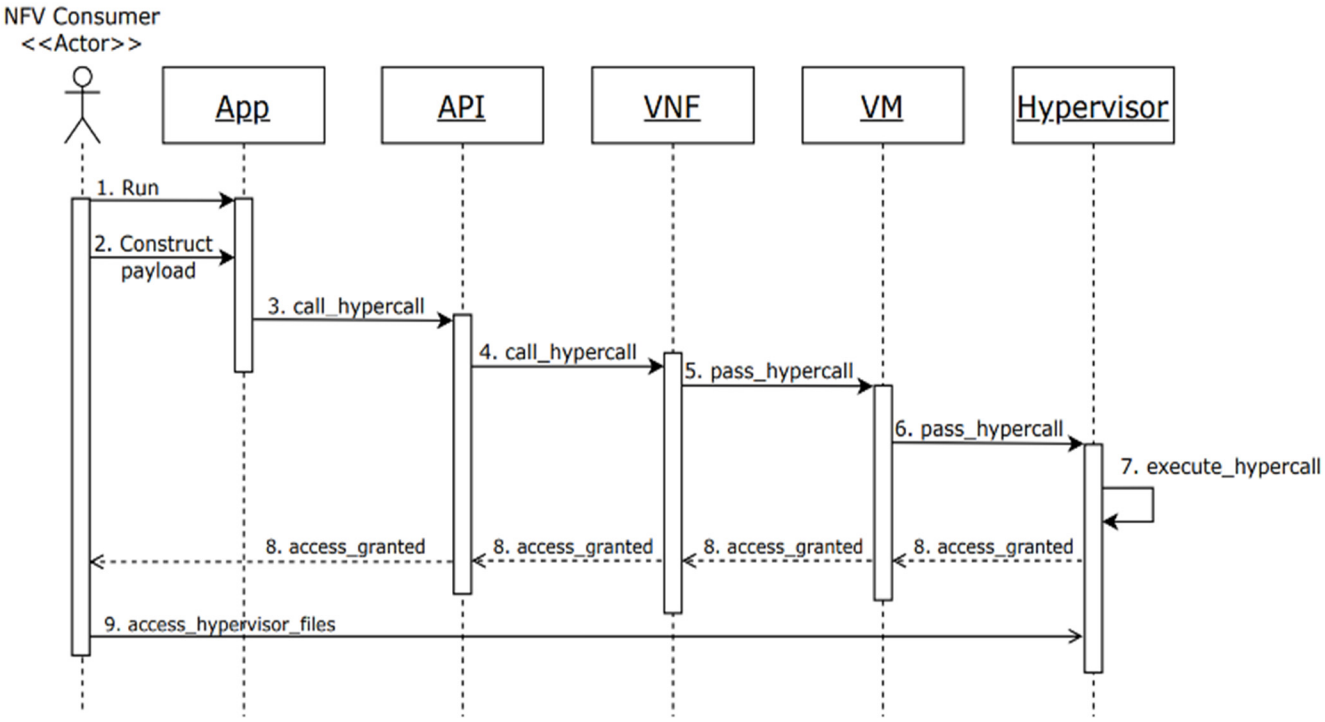

Figure 3. Sequence diagram for use case "Gain unauthorized access to hypervisor files by executing a malicious hypercall.".

Summary: The attacker, who is an NFV consumer, maliciously modifies the resource utilization data of the victim's VM, which affects the network service. In this misuse, the attacker modifies the weight value of the victim's VM from 256 to 64, which results in the victim having fewer resources than authorized. The dynamics of this misuse are shown in Figure 4.

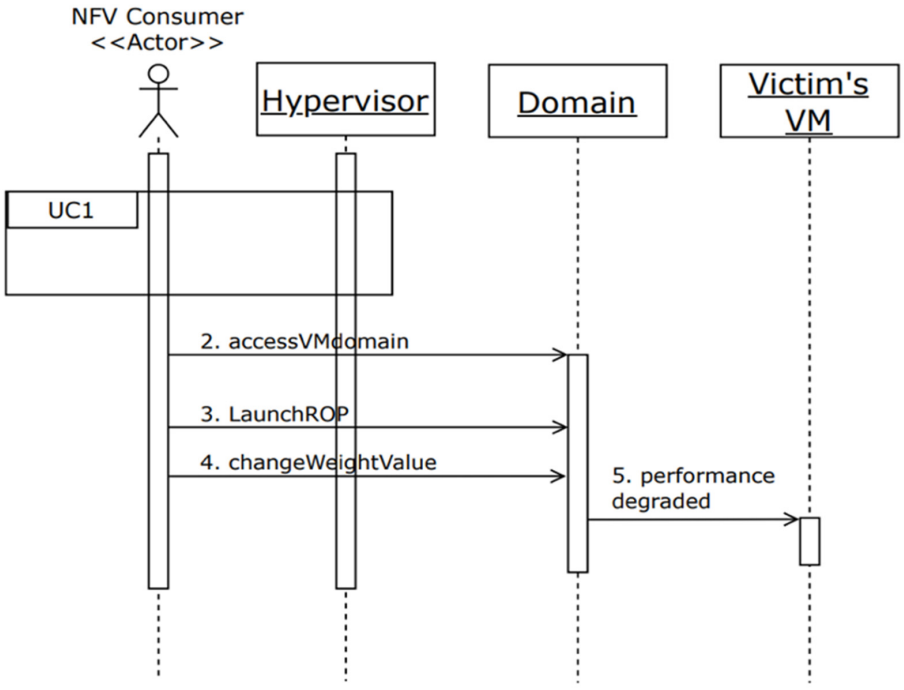

Figure 4. Sequence diagram for use case "Degrade the performance of the victim's network service".

Actor: NFV consumer (attacker)

Precondition: The attacker has a valid account and active network services.
Description:

1. The attacker performs Steps 1 to 9 of use case 1 (UC1). As a result, the attacker controls and accesses the hypervisor files.
2. The attacker accesses the domain structure of the victim's VM.
3. The attacker runs a return-oriented attack in order to modify the value of the VM's weight.
4. The attacker changes the value in the offset of the weight field in the domain structure.
5. The resulting modification of resource utilization data leads to degraded performance of the victim's VM, which then affects the network service (VNF) hosted in that VM.

Postcondition: The network service of the victim has been maliciously degraded.
Use case 3 (a misuse): Upgrade the performance of the attacker's network service.
Summary: The attacker, who is an NFV consumer, maliciously modifies the resource utilization data of their own VM in order to gain more resources. In this misuse, the attacker changes the value of the cap from 400 to 0, which results in the attacker having an unlimited cap. The dynamics of this misuse are shown in Figure 5.

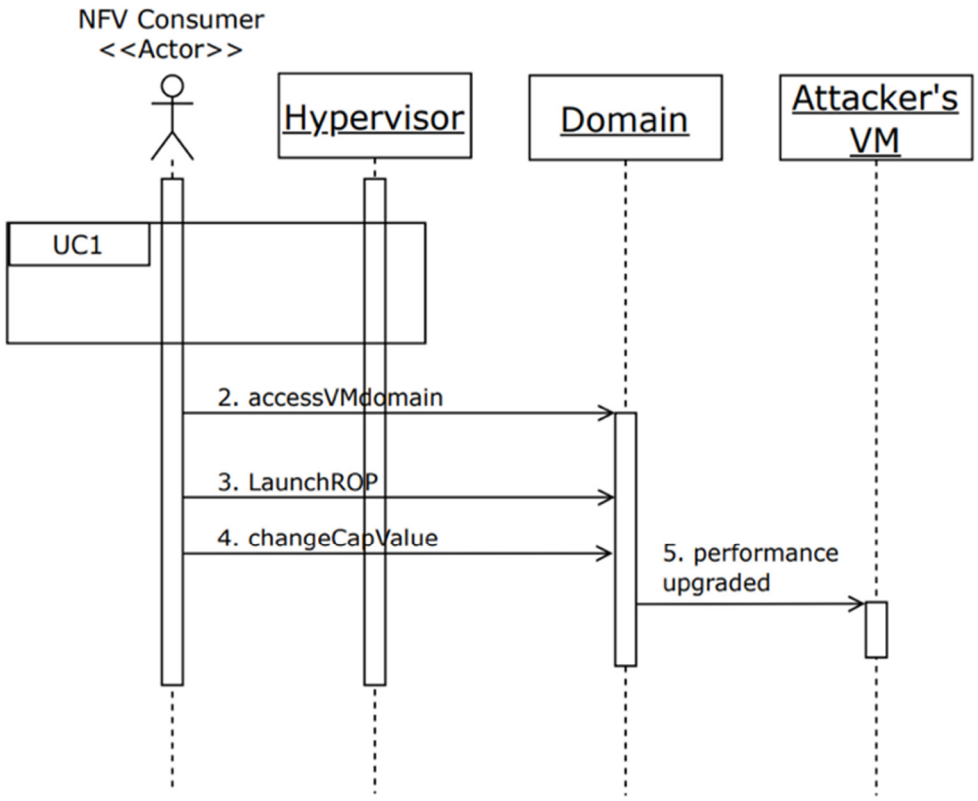

Figure 5. Sequence diagram for use case "Upgrade the performance of the attacker's network service".

Actor: NFV consumer (attacker)
Precondition: The attacker has a valid account and active network services.
Description:

1. The attacker performs Steps 1 to 9 of UC1 and gains control of and access to the hypervisor files.
2. The attacker accesses the domain structure of their own VM.
3. The attacker runs a return-oriented attack (ROP) in order to modify the cap value.
4. The attacker changes the value in the offset of the cap field in the domain structure.

5. The resulting modification of the resource utilization data leads to the upgraded performance of the attacker's VM.

Postcondition: The network service of the attacker has been maliciously upgraded.

3.7. Consequences

A successful attack leads to the following consequences:

1. The attacker is able to compromise the integrity of the NFV system by breaking into the hypervisor files [31,39,48].
2. The attacker will be able to carry out various activities related to misuse, such as migrating machines, stopping VNFs, and terminating network services [60,61].
3. NFV consumers, or victims, may receive less processing resources than they should receive, such as vCPU cap and weight, because the attacker has modified the resource utilization data [31].
4. The attacker can deploy a malicious payload, rendering out the resources of the system, causing a denial of service (DoS) [61].
5. The reputation of the targeted NFV provider could be negatively affected as they will appear to have security breaches [62].

Possible sources of failure include the following:

1. In the virtualization environment, different providers use different commercial hypervisors to run their virtualized services. For instance, in Xen hypervisor, the attacker must know the exact version of this hypervisor to ensure the success of such an attack.
2. This attack relies highly on the domain structure of VMs; the fields of domain structures may vary among commercial hypervisors.
3. The domain structures of VMs may also vary based on the configuration of the host hypervisor, which is accomplished by the administrator.
4. The method used in the attack scenario to identify the victim's VM is possible in an environment running a Xen hypervisor and may not work on other commercial hypervisors, such as VMware.
5. ROP attacks can be implemented in open-source hypervisors but would be difficult to implement in closed-source hypervisors because their data layout is not known [31].

3.8. Forensics

Evidence of such as attack can be discovered by the following actions:

1. NFV providers should keep logs of all hypercalls sent by VMs of all NFV consumers. Such logging will help to identify any malicious requests, whether sent by privileged or unprivileged users, that trigger the attacks.
2. NFV providers should apply security features that could help to detect evidence of ROP chain payload, such as ROPMENU [63]. This evidence involves artifacts that have been injected by malicious components.

3.9. Countermeasures

Attacks related to modifying non-control data can be mitigated using the following countermeasures:

1. Vendors of hypervisors should regularly patch hypervisor vulnerabilities that could lead to successful attacks. For instance, modifying non-control attacks can be mitigated using a vendor's patch in [64] if the attacker exploits this vulnerability [65].
2. Pointer Taintedness detection, an architectural technique that can mitigate both control and non-control data threats, should be implemented [66].
3. A data-oriented detection model should be used that analyzes the program source code to detect memory corruption on non-control data that could lead to illegal hypercalls [46].
4. NFV providers can apply several security tools to mitigate ROP attacks, such as G-Free [67], HyperCrop [68], HyperVerify [69], ROPecker [70], YARRA [29], and

PointGuard [71], as well as the hardware virtualization mechanism proposed in [72]. Stopping ROP attacks can stop the misuses in Figures 4 and 5.
5. Security schemes should be used to verify the authenticity and correctness of hypercalls, such as the Message Authentication Code (MAC) [73]. This defense can stop the attack in Figure 3.
6. Catching techniques, such as Hypercall Access Table (HAT), can be used to distinguish addresses of legitimate hypercalls and prevent calls from other locations not listed in the table [73]. However, this method will not mitigate attacks coming from authentic users. This defense also can stop the attack in Figure 3.
7. Trie Graph, a novel ROP attack defense methodology that monitors the integrity of control flow in the user space, should be applied [74]. This defense can stop the attacks in Figures 4 and 5.
8. ROPStop, a code reuse detection algorithm that can detect ROP exploits, should be applied [75]. This defense can stop the attacks in Figures 4 and 5.
9. RootkitDet, a defense system against the data modification threats at the kernel-level, should be applied [76].

3.10. Known Uses

The author in [45] explains the possibility of compromising the Xen hypervisor using the ROP attack. This scenario is applied in the cloud environment, in which NFV is an application of this virtualization environment. Therefore, in the context of NFV, the attack is possible in a way that compromises the integrity of resource utilization data and affects the performance of the victim's VMs [60]. Further, an integer overflow vulnerability in OpenSSH [77] has been discovered that allows attackers to write arbitrary data in any memory location; a detailed analysis of this vulnerability and its consequences are provided in [78–80]. Other incidents where non-control attacks have occurred are explained in [11–13].

3.11. Related Patterns

1. A Misuse Pattern for NFV based on Privilege Escalation: Shows the attackers the possibility to escalate the privilege of their VMs by exploiting a malicious hypercall [81].
2. NFV Virtual Machine Environment [22]: Describes the environment in which the hypervisor emulates resources, and creates and manages VMs for the purpose of providing virtualized network services.
3. Pattern for Network Function Virtualization [82]: Presents an abstract pattern for NFV architecture in which network services are created and deployed as cloud Software-as-a-Service (SaaS).
4. Virtual Machine Operating System Architecture (VMOS) [10]: Describes how VMs can run different operating systems, while remaining isolated from each other.
5. A Pattern for Network Function Virtualization Infrastructure (NFVI) [83]: Describes the architectural layer of NFV that contains the physical and virtual resources.

4. Use of Misuse Patterns

We presented a Security Reference Architecture for NFV [18]. In that work we showed three misuse patterns for three threats in NFV: privilege escalation [81], VM escape [84], and denial-of-service [85]. These misuse patterns were considered the first step toward building a security reference architecture (SRA) because they show what defenses (security patterns) can prevent the attack. As shown in the pattern diagram in Figure 6, a misuse pattern defines a threat to the NFV Reference Architecture (RA), and it can be mitigated by a security pattern. We can obtain an SRA by adding security patterns to the RA to control all the identified threats. In other words, an SRA is an RA where all its identified threats have been controlled by adding security patterns. If there are many threats, it is useful to perform a risk analysis where the probability of occurrence and the impact of the threat are evaluated. In our case, the defenses include several possible actions, any of which can

stop this attack. Figure 7 shows an extended version of the SRA in [18] showing where the defenses would be applied. The authenticator pattern in the hypervisor is used to verify the authenticity and correctness of the hypercall arguments coming from the guest VMs. This can mitigate the described attack by preventing the malicious hypercalls coming from compromised applications. More descriptions of the security patterns shown in Figure 7 can be found in [18]. Note that this is a partial SRA that can be completed by finding more threats and writing their corresponding misuse patterns.

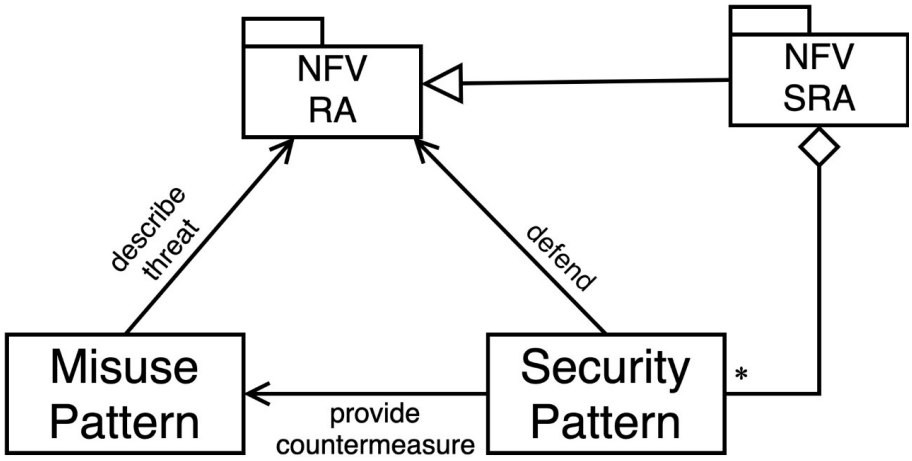

Figure 6. A pattern diagram for the NFV SRA [18].

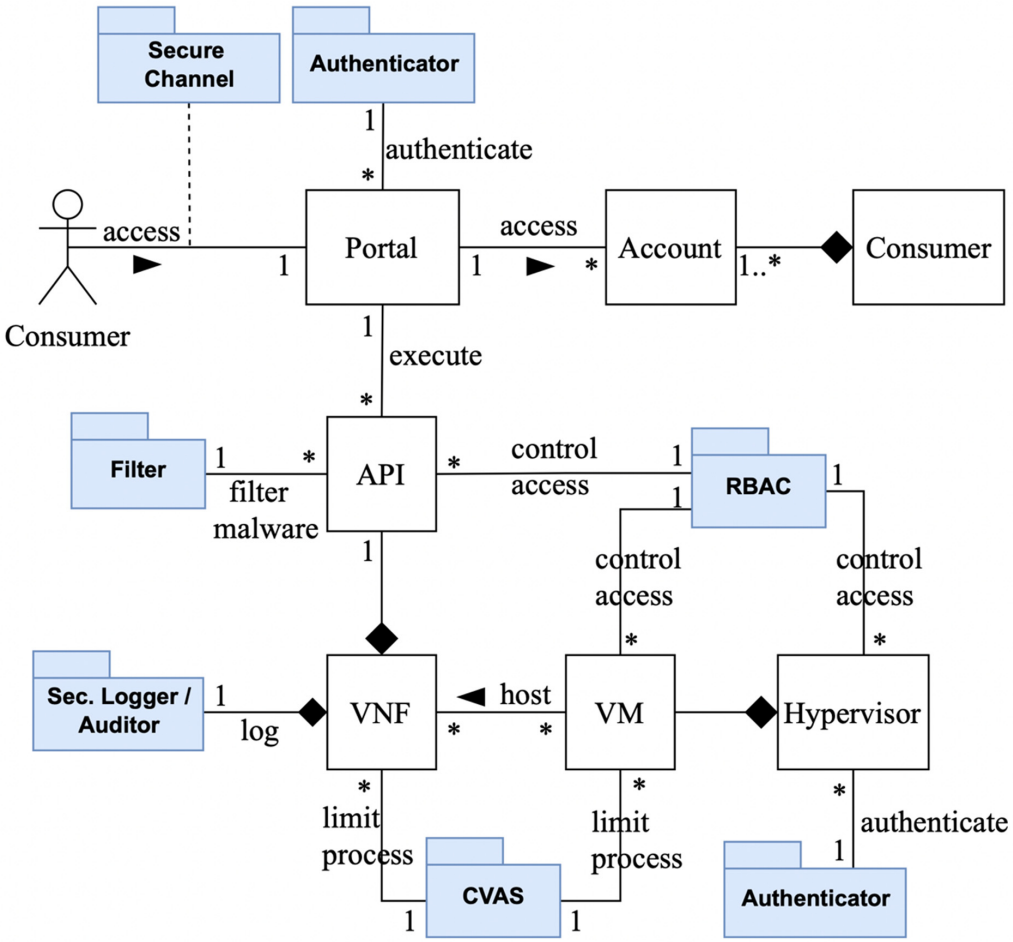

Figure 7. An extended version of the NFV SRA.

5. Related Work

The threat of modification of non-control data has been studied several times. The authors in [11] demonstrated how non-control data threats are realistic and can target real-world applications, such as FTB, SSH, Telnet, and HTTP servers. These applications have been reported by CERT as among the leading categories of vulnerable programs. In their scenarios, the attacks exploit buffer and integer overflow, heap corruption, and format string vulnerabilities. Further, the authors in [13] presented an expressive non-control data exploit using a data-oriented programming (DOP) technique, which uses only the data plane for malicious purposes without compromising the integrity of the control plane. They demonstrated that the attack is possible in nine real-world applications, eight of which showed possible arbitrary exploitations, and two of them were confirmed to build Turing-complete attacks.

Further, the authors in [12] showed different scenarios of modifying non-control data threats at the kernel level. One resultant scenario is a resource waste attack, which aims to degrade the performance of the applications running over the machine by creating artificial memory pressure and producing a large amount of unused memory spaces. The authors in [53] proposed a deep learning model to identify the security critical non-control data in program binaries; their neural model was effective as it achieved 87% accuracy.

The authors in [14] explained how a Control-Flow Binding (CFB) attack, a generalization of non-control data, is realistic in five real binaries. A CFB attack does not only modifies the data that affects the control-flow of the program as non-control data threats do, but also modifies the function pointer to point to a different system call. In the threat model they proposed, the memory corruption vulnerabilities can lead to two possible misuses, arbitrary code execution and information leakage.

The authors in [76] presented a threat model for modifying non-control data in a cloud environment. Their attack model was similar to what we showed earlier; they assume that the attacker exploits a vulnerability in the kernel and runs a malicious application in the VM to execute arbitrary code. As a result, the attacker will be able to take control of the VM and launch several attacks such as kernel-level data modifications and information leakage. Other works describe the threat of modifying non-control data in many systems [29,57,86,87].

However, based on the previously mentioned papers and readings of more works, none of the other works have used patterns as a precise way to analyze the threat of non-control data in a virtualized system. In fact, the abovementioned papers have explained different approaches to the modifying non-control data threats, but here we model the generic attack using patterns, which gives a global view and a more precise description of the threat.

Patterns have been proved as a convenient way to describe the threats in several environments. The authors in [16] used the misuse pattern to describe the threat of a distributed denial of service (DDoS) attack on Internet-of-Things (IoT) by constructing a botnet of infected IoT devices that could be used to attack a server. The authors in [15] used misuse patterns to describe the possible threats in the cloud environment. Misuse patterns have also been used to describe several possible threats in the NFV environment, including [81], VM escape [84], and DDoS [85].

6. Conclusions and Future Work

NFV is a network concept that leverages virtualization technologies obtained from cloud computing to provide reliable, flexible, and cost-efficient network services. However, these advantages come at the price of some security concerns. The hypervisor plays an important role in the virtualization environment, as it is responsible for emulating and sharing resources and creating and managing the VMs used to deliver network services [32]. In past work, we enumerated the possible threats that could jeopardize the virtualization environment in NFV [9]. We presented one of these threats here in the form of a misuse pattern.

Misuse patterns have been found to be useful in understanding threats and system vulnerabilities and designing secure systems, as well as performing forensics [10]. The idea of a misuse pattern has been used in [15,16,84,85], and we used it here to study the threat of modifying non-control data in the NFV environment. We showed how an attacker can exploit a vulnerability to modify the weight and cap values of a vCPU [45], which could lead to several misuses; we described three of them: gaining unauthorized access to hypervisor files by executing a malicious hypercall, degrading the performance of a victim's network service, and upgrade the performance of the attacker's network service. Other authors have shown different and complex approaches to describe the possible threats of modification of non-control data in the virtualized environment. We use Unified Modeling Language (UML) to build our misuse pattern, which gives a global view and a more precise description of the threat than block diagrams or word descriptions.

We are building a catalog of misuse patterns for NFV, and this pattern is part of it. Representing threats using misuse patterns can help us to identify the possible countermeasures for them in the form of security patterns. These security patterns can be used to build a security reference architecture (SRA) for NFV. We have started building the SRA [18], and currently, it includes the security patterns that could mitigate the previously identified misuses [81,84,85]. The description of the modifying non-control threat pattern that we presented here will be part of our SRA.

Further studies are needed to understand the other threats that may compromise non-control data. We think non-control data related to the policies of network services and service level agreements (SLAs) need to be investigated because compromising the integrity of these types of data will have a major effect on the network service lifecycle.

Funding: This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Project No. GRANT272].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Many thanks to the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, for supporting this research. I also thank Eduardo B. Fernandez for his valuable comments that helped improve this paper.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. A Template for Misuse Patterns

- Name: Indicates the name of the misuse pattern, and it should correspond to a generic name that is given to a specific type of threat in standard attack repositories.
- Intent: Describes the problem it solves for an attacker.
- Context: Describes the environment, including the conditions, in which the attack may occur.
- Problem: Shows, from the attacker's perspective, how to find a way to attack a system; the forces indicate what vulnerabilities can be exploited to accomplish the attack.
- Solution: Shows the solution to the attacker's problem; it shows how the attack reaches its objectives and its expected results. The solution section includes structure and dynamics.
    - Structure: Shows in a UML class diagram all the components involved in the attack.
    - Dynamics: Shows in UML sequence or collaboration diagrams the exchange of messages needed between system components to accomplish the attack.
- Consequences: Discusses the advantages and drawbacks of the attack from the attacker's perspective.

- Forensics: Describes what information can be obtained to improve forensic analysis by tracing the attack to its source.
- Countermeasures: Describes the security measures necessary to prevent, mitigate, or trace the attack.
- Known uses: Lists the security incidents where the attack has already occurred.

References

1. Sinh, D.C.; Le, L.V.; Lin, B.S.P.; Tung, L.P. SDN/NFV—A New Approach of Deploying Network Infrastructure for IoT. In Proceedings of the 27th Wireless and Optical Communication Conference, WOCC, Hualien, Taiwan, 30 April–1 May 2018; pp. 1–5.
2. Masutani, H.; Nakajima, Y.; Kinoshita, T.; Hibi, T.; Takahashi, H.; Obana, K.; Shimano, K.; Fukui, M. Requirements and Design of Flexible NFV Network Infrastructure Node Leveraging SDN/OpenFlow. In Proceedings of the 2014 International Conference on Optical Network Design and Modeling, Stockholm, Sweden, 19–22 May 2014; pp. 258–263.
3. Manzalini, A.; Italia, T.; Roberto Saracco, I.; Labs, E.; Cagatay Buyukkoc, I.; Gladisch, A.; Fukui, M.; Shen, W.; Eliezer Dekel, J.; David Soldani, I.; et al. Software-Defined Networks for Future Networks and Services Main Technical Challenges and Business Implications. *White Paper Based on the IEEE Workshop SDN4FNS*. 2014. Available online: https://discovery.ucl.ac.uk/id/eprint/10043677/1/White%20Paper%20IEEE%20SDN4FNS-FinalVersion.pdf (accessed on 18 June 2022).
4. Yoshida, M.; Shen, W.; Kawabata, T.; Minato, K.; Imajuku, W. MORSA: A Multi-Objective Resource Scheduling Algorithm for NFV Infrastructure. In Proceedings of the 16th Asia-Pacific Network Operations and Management Symposium, Hsinchu, Taiwan, 17–19 September 2014; pp. 1–6.
5. Bouras, C.; Ntarzanos, P.; Papazois, A. Cost Modeling for SDN/NFV Based Mobile 5G Networks. In Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, Lisbon, Portugal, 18–20 October 2016; pp. 56–61.
6. Yoon, M.S.; Kamal, A.E. NFV Resource Allocation Using Mixed Queuing Network Model. In Proceedings of the 2016 IEEE Global Communications Conference, GLOBECOM, Washington, DC, USA, 4–8 December 2016; pp. 1–6.
7. Lal, S.; Taleb, T.; Dutta, A. NFV: Security Threats and Best Practices. *IEEE Commun. Mag.* **2017**, *55*, 211–217. [CrossRef]
8. Yang, W.; Fung, C. A Survey on Security in Network Functions Virtualization. In Proceedings of the IEEE NetSoft Conference and Workshops: Software-Defined Infrastructure for Networks, Clouds, IoT and Services, Seoul, Korea, 6–10 June 2016; pp. 15–19.
9. Alwakeel, A.M.; Alnaim, A.K.; Fernandez, E.B. A Survey of Network Function Virtualization Security. In Proceedings of the IEEE Southeastcon, St. Petersburg, FL, USA, 19–22 April 2018; pp. 1–8.
10. Fernandez, E.B. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns*; J. Wiley & Sons: Hoboken, NJ, USA, 2013; ISBN 9781119998945.
11. Chen, S.; Xu, J.; Sezer, E.C.; Gauriar, P.; Iyer, R.K. Non-Control-Data Attacks Are Realistic Threats. In Proceedings of the 14th Conference on USENIX Security Symposium, Baltimore, MD, USA, 31 July–5 August 2005.
12. Baliga, A.; Kamat, P.; Iftode, L. Lurking in the Shadows: Identifying Systemic Threats to Kernel Data (Short Paper). In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 246–251.
13. Hu, H.; Shinde, S.; Adrian, S.; Chua, Z.L.; Saxena, P.; Liang, Z. Data-Oriented Programming: On the Expressiveness of Non-Control Data Attacks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 969–986.
14. Carlini, N.; Barresi, A.; Payer, M.; Wagner, D.A.; Gross, T. Control-Flow Bending: On the Effectiveness of Control-Flow Integrity. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 12–14 August 2015; pp. 161–176.
15. Hashizume, K.; Yoshioka, N.; Fernandez, E.B. Misuse Patterns for Cloud Computing. In Proceedings of the 2nd Asian Conference on Pattern Languages of Programs—AsianPLoP '11, Tokyo, Japan, 5–8 October 2011; pp. 1–6.
16. Syed, M.H.; Fernandez, E.B.; Moreno, J. A Misuse Pattern for DDoS in the IoT. In Proceedings of the 23rd European Conference on Pattern Languages of Programs, Irsee, Germany, 4–8 July 2018; ACM: New York, NY, USA, 2018; pp. 1–5.
17. Pelaez, J.C.; Fernandez, E.B.; Larrondo-Petrie, M.M.; Wieser, C. Misuse Patterns in VoIP. In Proceedings of the 14th Conference on Pattern Languages of Programs—PLOP '07, Monticello, IL, USA, 5–8 September 2007; ACM: New York, NY, USA; pp. 1–13.
18. Alnaim, A.K.; Alwakeel, A.M.; Fernandez, E.B. Towards a Security Reference Architecture for NFV. *Sensors* **2022**, *22*, 3750. [CrossRef] [PubMed]
19. Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerland, P.; Stal, M. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*; Wiley: Hoboken, NJ, USA, 1996; Volume 1, ISBN 978-0-471-95869-7.
20. ETSI. *Network Functions Virtualisation (NFV); Architectural Framework*. 2014. Available online: https://cdn.standards.iteh.ai/samples/43827/5288dd7aff4b4de6a4a63a5034c00168/ETSI-GS-NFV-002-V1-2-1-2014-12-.pdf (accessed on 18 June 2022).
21. Chandramouli, R. *Security Recommendations for Hypervisor Deployment on Servers-NIST Special Publication 800-125A*; 2018. Available online: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-125A.pdf (accessed on 18 June 2022).
22. Alnaim, A.K.; Alwakeel, A.M.; Fernandez, E.B. A Pattern for an NFV Virtual Machine Environment. In Proceedings of the 13th Annual IEEE International Systems Conference, Orlando, FL, USA, 8–11 April 2019; pp. 1–6.

23. Syed, M.H.; Fernandez, E.B. A Reference Architecture for the Container Ecosystem. In Proceedings of the ACM International Conference Proceeding Series, Hamburg, Germany, 27–30 August 2018; pp. 1–6.
24. Fernandez, E.B.; Yoshioka, N.; Washizaki, H.; Syed, M.H. Modeling and Security in Cloud Ecosystems. *Future Internet* **2016**, *8*, 13. [CrossRef]
25. Sulatycki, R.; Fernandez, E.B. A Threat Pattern for the "Cross-Site Scripting (XSS)" Attack. In Proceedings of the 22nd Conference on Pattern Languages of Programs, Pittsburgh, PA, USA, 24–26 October 2015.
26. Cybersecurity and Infrastructure Security Agency (CISA). CERT Security Advisories CISA. Available online: https://www.cisa.gov/uscert/ics/advisories (accessed on 16 January 2022).
27. Microsoft. Microsoft Security Bulletins. Available online: https://docs.microsoft.com/en-us/security-updates/securitybulletins/securitybulletins (accessed on 18 June 2022).
28. Abadi, M.; Budiu, M.; Erlingsson, Ú.; Ligatti, J. Control-Flow Integrity Principles, Implementations, and Applications. *ACM Trans. Inf. Syst. Secur. TISSEC* **2009**, *13*, 1–40. [CrossRef]
29. Schlesinger, C.; Pattabiraman, K.; Swamy, N.; Walker, D.; Zorn, B. Modular Protections against Non-Control Data Attacks. In Proceedings of the IEEE Computer Security Foundations Symposium, Cernay-la-Ville, France, 27–29 June 2011; pp. 131–145.
30. Sotirov, A. Modern Exploitation and Memory Protection Bypasses. 2009. Available online: https://www.usenix.org/conference/usenixsecurity09/technical-sessions/presentation/sotirov (accessed on 18 June 2022).
31. Ding, B.; He, Y.; Wu, Y.; Yu, J. Systemic Threats to Hypervisor Non-Control Data. *IET Inf. Secur.* **2013**, *7*, 349–354. [CrossRef]
32. ETSI. *Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain*. 2015. Available online: https://www.etsi.org/deliver/etsi_gs/nfv-inf/001_099/004/01.01.01_60/gs_nfv-inf004v010101p.pdf (accessed on 18 June 2022).
33. Garfinkel, T.; Rosenblum, M. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In Proceedings of the Annual Network and Distributed Systems Security Symp, San Diego, CA, USA, 6 February 2003; pp. 191–206.
34. Jiang, X.; Wang, X.; Xu, D. Stealthy Malware Detection through VMM-Based "out-of-the-Box" Semantic View Reconstruction. In Proceedings of the ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 31 October–2 November 2007; pp. 128–138.
35. Payne, B.D.; Carbone, M.; Sharif, M.; Lee, W. Lares: An Architecture for Secure Active Monitoring Using Virtualization. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 18–22 May 2008; pp. 233–247.
36. Litty, L.; Andrés Lagar-Cavilla, H.; Lie, D. Hypervisor Support for Identifying Covertly Executing Binaries. In Proceedings of the USENIX Security Symp, San Jose, CA, USA, 28 July–1 August 2008; p. 258.
37. SynopSys Black Duck Open Hub-Xen Project (Hypervisor). Available online: https://www.openhub.net/p/xenproject-hypervisor/analyses/latest/languages_summary (accessed on 18 June 2022).
38. SynopSys Black Duck Open Hub-KVM. Available online: https://www.openhub.net/p/kvm/analyses/latest/languages_summary (accessed on 18 June 2022).
39. Perez-Botero, D.; Szefer, J.; Lee, R.B. Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers. In Proceedings of the International Workshop on Security in Cloud Computing—Cloud Computing '13, Hangzhou, China, 8 May 2013; pp. 3–10.
40. NIST. National Vulnerability Database—CVE-2011-1898. Available online: https://nvd.nist.gov/vuln/detail/CVE-2011-1898 (accessed on 18 June 2022).
41. NIST. National Vulnerability Database—CVE-2021-36148. Available online: https://nvd.nist.gov/vuln/detail/CVE-2021-36148 (accessed on 18 June 2022).
42. NIST. National Vulnerability Database—CVE-2021-38923. Available online: https://nvd.nist.gov/vuln/detail/CVE-2021-38923 (accessed on 18 June 2022).
43. Milenkoski, A.; Payne, B.D.; Antunes, N.; Vieira, M.; Kounev, S. Experience Report: An Analysis of Hypercall Handler Vulnerabilities. In Proceedings of the International Symposium on Software Reliability Engineering, ISSRE, Naples, Italy, 3–6 November 2014; pp. 100–111.
44. Riddle, A.R.; Chung, S.M. A Survey on the Security of Hypervisors in Cloud Computing. In Proceedings of the IEEE 35th International Conference on Distributed Computing Systems Workshops, ICDCSW, Columbus, OH, USA, 29 June–2 July 2015; pp. 100–104.
45. Ding, B.; Wu, Y.; He, Y.; Tian, S.; Guan, B.; Wu, G. Return-Oriented Programming Attack on the Xen Hypervisor. In Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security, Prague, Czech Republic, 20 August 2012; pp. 479–484.
46. Demay, J.C.; Totel, E.; Tronel, F. SIDAN: A Tool Dedicated to Software Instrumentation for Detecting Attacks on Non-Control-Data. In Proceedings of the 4th International Conference on Risks and Security of Internet and Systems, CRiSIS, Toulouse, France, 19–22 October 2009; pp. 51–58.
47. Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. Xen and the Art of Virtualization. In Proceedings of the ACM symposium on Operating systems Principles, Bolton Landing, NY, USA, 19–22 October 2003; p. 177.
48. Wojtczuk, R. Subverting the Xen Hypervisor. *Black Hat USA* **2008**, *2008*, 2.
49. Jansen, W.A. Cloud Hooks: Security and Privacy Issues in Cloud Computing. In Proceedings of the 44th Hawaii International Conference on System Sciences, Kauai, HI, USA, 4–7 January 2011; pp. 1–10.

50. NIST. National Vulnerability Database—CVE-2014-1893. Available online: https://nvd.nist.gov/vuln/detail/CVE-2014-1893 (accessed on 18 June 2022).
51. NIST. National Vulnerability Database—CVE-2012-6032. Available online: https://nvd.nist.gov/vuln/detail/CVE-2012-6032 (accessed on 18 June 2022).
52. Zhang, G.; Li, Q.; Chen, Z.; Zhang, P. Defending Non-Control-Data Attacks Using Influence Domain Monitoring. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 3888–3910. [CrossRef]
53. Wang, Z.; Wang, H.; Hu, H.; Liu, P. Identifying Non-Control Security-Critical Data in Program Binaries with a Deep Neural Model 2021. Available online: https://arxiv.org/pdf/2108.12071.pdf (accessed on 18 June 2022).
54. ETSI. *Network Functions Virtualisation (NFV); NFV Security; Problem Statement*. 2014. Available online: https://www.etsi.org/deliver/etsi_gs/nfv-sec/001_099/001/01.01.01_60/gs_nfv-sec001v010101p.pdf (accessed on 18 June 2022).
55. NIST. National Vulnerability Database—CVE-2011-1583. Available online: https://nvd.nist.gov/vuln/detail/CVE-2011-1583 (accessed on 18 June 2022).
56. Abels, T.; Dhawan, P.; Chandrasekaran, B. *An Overview of Xen Virtualization*. 2005. Available online: https://courses.cs.vt.edu/~{}cs5204/fall07-kafura/Papers/Virtualization/Xen-ShortOverview.pdf (accessed on 18 June 2022).
57. Hu, H.; Chua, Z.L.; Adrian, S.; Saxena, P.; Liang, Z. Automatic Generation of Data-Oriented Exploits. In Proceedings of the 24th USENIX Conference on Security Symposium, Washington, DC, USA, 12–14 August 2015; pp. 177–192.
58. Checkoway, S.; Davi, L.; Dmitrienko, A.; Sadeghi, A.-R.; Shacham, H.; Winandy, M. Return-Oriented Programming without Returns. In Proceedings of the the 17th ACM Conference on Computer and Communications Security—CCS '10, Chicago, IL, USA, 4–8 October 2010; pp. 559–572.
59. Carlini, N.; Wagner, D. ROP Is Still Dangerous: Breaking Modern Defenses. In Proceedings of the 23rd USENIX conference on Security Symposium, San Diego, CA, USA, 20–22 August 2014; pp. 395–399.
60. Reynaud, F.; Aguessy, F.-X.; Bettan, O.; Bouet, M.; Conan, V. Attacks against Network Functions Virtualization and Software-Defined Networking: State-of-the-Art. In Proceedings of the IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea, 6–10 June 2016; pp. 471–476.
61. ETSI. *Network Functions Virtualisation (NFV); NFV Security; Security and Trust Guidance*. 2014. Available online: https://www.etsi.org/deliver/etsi_gs/nfv-sec/001_099/003/01.01.01_60/gs_nfv-sec003v010101p.pdf (accessed on 18 June 2022).
62. Alshammari, S.T.; Albeshri, A.; Alsubhi, K. Building a Trust Model System to Avoid Cloud Services Reputation Attacks. *Egypt. Inform. J.* **2021**, *22*, 493–503. [CrossRef]
63. Graziano, M.; Eurecom, D.B.; Zidouemba, A. ROPMEMU: A Framework for the Analysis of Complex Code-Reuse Attacks. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, 30 May–3 June 2016; pp. 47–58.
64. Xen Project Xen Security Advisory. Available online: http://old-list-archives.xenproject.org/archives/html/xen-devel/2011-05/msg00483.html (accessed on 15 November 2021).
65. NIST. National Vulnerability Database—CVE-2018-15471. Available online: https://nvd.nist.gov/vuln/detail/CVE-2018-15471 (accessed on 8 April 2022).
66. Chen, S.; Xu, J.; Nakka, N.; Kalbarczyk, Z.; Iyer, R.K. Defeating Memory Corruption Attacks via Pointer Taintedness Detection. In Proceedings of the International Conference on Dependable Systems and Networks, Yokohama, Japan, 28 June–1 July 2005; pp. 378–387.
67. Onarlioglu, K.; Bilge, L.; Lanzi, A.; Balzarotti, D.; Kirda, E. G-Free: Defeating Return-Oriented Programming through Gadget-Less Binaries. In Proceedings of the 26th Annual Computer Security Applications Conference on—ACSAC '10, Austin, TX, USA, 6–10 December 2010; ACM Press: New York, NY, USA; p. 49.
68. Jiang, J.; Jia, X.; Feng, D.; Zhang, S.; Liu, P. HyperCrop: A Hypervisor-Based Countermeasure for Return Oriented Programming. In Proceedings of the International Conference on Information and Communications Security, Beijing, China, 23–26 November 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 360–373.
69. Ding, B.; He, Y.; Wu, Y.; Lin, Y. HyperVerify: A VM-Assisted Architecture for Monitoring Hypervisor Non-Control Data. In Proceedings of the IEEE Seventh International Conference on Software Security and Reliability Companion, Gaithersburg, MD, USA, 18–20 June 2013; pp. 26–34.
70. Cheng, Y.; Zhou, Z.; Yu, M.; Ding, X.; Deng, R.H. ROPecker: A Generic and Practical Approach For Defending Against ROP Attacks. In Proceedings of the 21st Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014; pp. 1–14.
71. Cowan, C.; Beattie, S.; Johansen, J.; Wagle, P. PointGuard: Protecting Pointers From Buffer Overflow Vulnerabilities. In Proceedings of the 12th conference on USENIX Security Symposium, Washington, DC, USA, 4–8 August 2003; pp. 91–104.
72. Shuo, T.; Yeping, H.; Baozeng, D. Prevent Kernel Return-Oriented Programming Attacks Using Hardware Virtualization. In Proceedings of the International Conference on Information Security Practice and Experience, Hangzhou, China, 9–12 April 2012; pp. 289–300.
73. Hoang, C.; Hoang, C.; Le, H. Protecting Xen Hypercalls Intrusion Detection/Prevention in a Virtualization Environment. Master Thesis, The University of British Columbia, Vancouver, BC, Canada, 2009.
74. Zhu, A.Y.C.; Yan, W.Q.; Sinha, R. ROP Defense Using Trie Graph for System Security. *Int. J. Digit. Crime Forensics IJDCF* **2021**, *13*, 1–12. [CrossRef]

75. Jacobson, E.R.; Bernat, A.R.; Williams, W.R.; Miller, B.P. Detecting Code Reuse Attacks with a Model of Conformant Program Execution. In Proceedings of the International Symposium on Engineering Secure Software and Systems, Munich, Germany, 26–28 February 2014; pp. 1–18.
76. Zhang, L.; Shetty, S.; Liu, P.; Jing, J. RootkitDet: Practical End-to-End Defense against Kernel Rootkits in a Cloud Environment. *Eur. Symp. Res. Comput. Secur.* **2014**, *8713*, 475–493. [CrossRef]
77. NIST. National Vulnerability Database—CVE-2001-0144. Available online: https://nvd.nist.gov/vuln/detail/CVE-2001-0144 (accessed on 18 June 2022).
78. Pekka, K.; Kalle, L. SSHD CRC32 Compensation Attack Detector Vulnerability Explained. Available online: https://www.youngsam.net/entry/SSH1-remote-root-exploit (accessed on 27 January 2022).
79. Dittrich, D.A. Analysis of SSH Crc32 Compensation Attack Detector Exploit. Available online: https://newtotse.com/oldtotse/en/hack/hack_attack/162684.html (accessed on 18 June 2022).
80. Starzetz, P. "SSH1 CRC32 Vulnerability Analysis. Available online: https://packetstormsecurity.com/files/24347/ssh1.crc32.txt.html (accessed on 27 January 2022).
81. Alnaim, A.K.; Alwakeel, A.M.; Fernandez, E.B. A Misuse Pattern for NFV Based on Privilege Escalation. In Proceedings of the 8th Asian Conference on Pattern Languages of Programs, Tokyo, Japan, 20–22 March 2019.
82. Fernandez, E.B.; Hamid, B. A Pattern for Network Functions Virtualization. In Proceedings of the 20th European Conference on Pattern Languages of Programs—EuroPLoP '15, Kaufbeuren, Germany, 8–12 July 2015; ACM Press: New York, NY, USA; pp. 1–9.
83. Alwakeel, A.M.; Alnaim, A.K.; Fernandez, E.B. A Pattern for Network Function Virtualization Infrastructure (NFVI). In Proceedings of the 26th PLoP'19, Ottawa, ON, Canada, 7–10 October 2019; pp. 1–9.
84. Alnaim, A.K.; Alwakeel, A.M.; Fernandez, E.B. A Misuse Pattern for Compromising VMs via Virtual Machine Escape in NFV. In Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019), Canterbury, UK, 26–29 August 2019; pp. 1–6.
85. Alnaim, A.K.; Alwakeel, A.M.; Fernandez, E.B. A Misuse Pattern for Distributed Denial-of-Service Attack in Network Function Virtualization. In Proceedings of the PLoP '19: Pattern Languages of Programs Conference, Ottawa, ON, Canada, 7–10 October 2019; pp. 1–10.
86. Díez-Franco, I.; Santos, I. Data Is Flowing in the Wind: A Review of Data-Flow Integrity Methods to Overcome Non-Control-Data Attacks. *Adv. Intell. Syst. Comput.* **2017**, *527*, 536–544. [CrossRef]
87. Vogl, S.; Gawlik, R.; Garmany, B.; Kittel, T.; Pfoh, J.; Eckert, C.; Holz, T. Dynamic Hooks: Hiding Control Flow Changes within Non-Control Data. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, 20–22 August 2014; pp. 813–823.