Metaverse Basic and Applied Research. 2023; 2:37
doi: 10.56294/mr202337

ORIGINAL

Securing Virtual Architecture of Smartphones based on Network Function Virtualization

Seguridad de la arquitectura virtual de los teléfonos inteligentes basada en la virtualización de funciones de red

Hayder A. Nahi[1], Mustafa Asaad Hasan[2], Ali Hussein Lazem[2], Mohammed Ayad Alkhafaji[3]

[1]Al-Qasim Green Univrsity. Computer Center. Babylon, Iraq.
[2]University of Thi-Qar. Iraq.
[3]National University of Science and Technology. College of Technical Engineering, Department of Medical Devices Technologies. Dhi Qar, Iraq.

Cite as: Nahi HA, Hasan MA, Lazem AH, Alkhafaji MA. Securing Virtual Architecture of Smartphones based on Network Function Virtualization. Metaverse Basic and Applied Research. 2023; 2:37. https://doi.org/10.56294/mr202337

Submitted: 24-02-2023 **Revised:** 01-05-2023 **Accepted:** 14-07-2023 **Published:** 15-07-2023

Editor: Lic. Mabel Cecilia Bonardi

ABSTRACT

One of the most difficult parts of Network Function Virtualization (NFV) installations is security. The NFV environment is a large-scale, software-driven one with a variety of components. Network topologies and traffic flows are continuously and managed to change. Such complexity necessitates a comprehensive security framework that permits automatic and to manage changeable network conditions, a quick response is required with the least amount of manual involvement. This paper introduced many solutions for securing the NFV environment from attacks such as (Specter and DoS) that attack parts of this architecture based on some experiments. Applied NFV on an operating system of smartphones (Android). We tested some attacks on the device and then on some of the layers in the architecture. We obtain new and obvious results, by comparison, to traditional and updated NFV architecture. Also, update the NFV architecture using vCenter/ESX and Hyper-V being two important terms in security After adding the necessary algorithms to protect the NFV architecture, we noticed about 128 hours to hack a 1,4 megabyte (WinRAR) file, while the same file and the same size needed 126 hours to reach the root without the algorithms used to protect the architecture.

Keywords: Network Function Virtualization; Hypervisor; Specter; DoS; Smartphones.

RESUMEN

Una de las partes más difíciles de las instalaciones de Virtualización de Funciones de Red (NFV) es la seguridad. El entorno de NFV es uno de gran escala, impulsado por software con una variedad de componentes. Las topologías de red y los flujos de tráfico cambian y se administran continuamente. Tal complejidad requiere un marco de seguridad integral que permita una respuesta rápida y la capacidad de manejar las condiciones de la red cambiantes, con la menor cantidad posible de intervención manual. Este documento presenta muchas soluciones para asegurar el entorno de NFV contra ataques como Specter y DoS, que atacan partes de esta arquitectura basándose en algunos experimentos. Se aplicó NFV en un sistema operativo de teléfonos inteligentes (Android). Probamos algunos ataques en el dispositivo y luego en algunas de las capas de la arquitectura. Obtenemos resultados nuevos y evidentes, en comparación con la arquitectura NFV tradicional y actualizada. También actualizamos la arquitectura de NFV utilizando vCenter/ESX e Hyper-V, siendo dos términos importantes en seguridad. Después de agregar los algoritmos necesarios para proteger la arquitectura de NFV, notamos que se necesitaron alrededor de 128 horas para hackear un archivo de 1.4 megabytes (WinRAR), mientras que el mismo archivo y tamaño necesitaron 126 horas para alcanzar la raíz sin los algoritmos utilizados para proteger la arquitectura.

Palabras clave: Virtualización de Funciones de Red, Hypervisor; Specter; DoS; Smartphones.

INTRODUCTION

Telecommunications service providers (TSPs) today must proportionately and continuously buy, store, and run new existing hardware due to the growing variety and data rates via consumers. For TSPs, it results in substantial expenditure costs. By separating the functions that run on physical networking devices from the device itself, NFV [1,2,3,4] has been presented as a modern technology to develop, implement, and control network infrastructure at significantly lower costs. Further particularly, NFV makes use of virtualization technologies to deliver network functions (NFs) by executing software on large volume servers, gateways, and memory that are industry standards.[3] The primary benefit of NFV is the realization of software-based NFs rather of hardware appliances, such as virtualized firewalls in addition to gateways.

The benefits of NFV over conventional network architectures are numerous,[5] a more effective network architecture and resource allocation, cheaper equipment costs, higher operating performance and operational efficiency, more scalable function deployment and dynamic implementation, and lower energy usage. But NFV might have a number of security problems. For instance, elements of the NFV framework such hypervisors and orchestrators might be exposed to security from the risks. New security flaws could be introduced via shared storage and networking.[1] Additionally, multiple suppliers are likely to offer different hypervisors, hardware, and VNFs, which complicates integration and creates security gaps.[5]

The security issues with NFV have been addressed using a variety of strategies.[6,7,8,9] For instance, the NFV ISG proposes associated technologies to provide security and trust for NFV as well as guidelines for ensuring security in the NFV's exterior operating context.[10,11] Alcatel-Lucent discussed the corresponding mitigating techniques and described the security threats that now exist in NFV. Huawei emphasized the importance of offering efficient security monitoring to find risks and prevent attacks.[12]

The main of the key features of cloud computing is virtualization, that permits companies to expand or virtualize the activities of their resources in order to increase a reliability of the system work. In other aspects, virtualization enables the use of several users or even multiple businesses to share one physical instance of a resource as a virtual machine.[13] Hypervisor, meanwhile, is used to separate and manage the various virtual machines from the actual computing devices in order to manage the use of shared resources.[14]

Through a straightforward programmable plug-in and broker interface, the Management and Network Orchestration (MANO) platform implements VNF and application awareness.[15] It is an expandable, scalable solution. It offers a complete solution for managing network and service functions across various infrastructure platforms.

A new strategy for interactive computing is known as "software-defined networking" (SDN), which describes the ability to establish, manage, modify, and administer network activities automatically over open interfaces. Through the establishment of an abstraction for the data forwarding plane and the subsequent separation of it from the control plane, SDN highlights the importance of software in managing networks.[16]

Virtual network functions (VNFs) are software-based versions of network operations that were previously carried out by specialized and proprietary hardware devices. VNFs are designed to replace such devices by running on cost-effective hardware while performing specific network and network security functions. Routers, firewalls, the Domain Name System (DNS), load balancing, caching, and network address translation are some of the tools that are utilized for these activities by both businesses and network service providers.[17]

This paper particularly concentrates on the security part of NFV. Extra precisely, the goals of this paper are to provide results of attacking parts of NFV architecture, then re-attacked after securing these parts by many algorithms. We generated two attacks are spectre and DoS the first one tested on the hardware devices and second one on the hypervisor layer.

NFV Architecture Security Challenges with Solutions

Many advantages like lower costs and less operational influence are made possible with NFV. However, advantages do come with problems.[18,19] Authentication and authorisation of users and tenants are fundamental issues with NFV security.[10] The security aspect is discusses in[11] via three scenarios for NFV security: security within VNFs, security across VNFs, and security external to VNFs. NFV's biggest challenges are security and software management. [12] A barrier for implementing NFV in telecommunication networks and mobile networks is how to solve the security issues posed by hypervisors, data communication, and APIs.[5] As shown in figure 1, NFV architectural framework includes many functional parts.

Physical Layer

The Hardware is at the bottom. For CSPs, any vulnerability that damages this layer—such as those from Spectre and Meltdown flaws—and exposes data privacy issues might be disastrous. The source of trust function a collection of hardware and software security modules ought to be turned on the server in order to secure this. A better solution for the operating system to be executed is established by this base of trust.

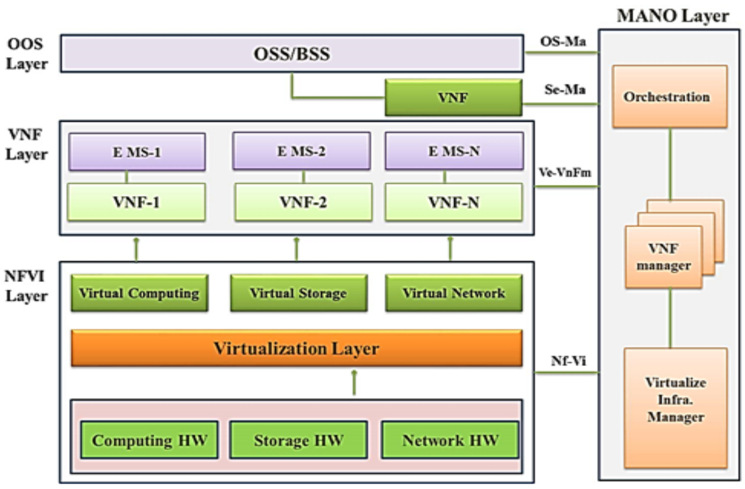


Figure 1. NFV architecture[3]

Hypervisor Layer

The software called a hypervisor enables numerous guest virtual computers to run in parallel on the same host. Virtualization security is strongly reliant on the individual security within each part, including the hypervisor, host operating system, guest operating system, applications, and storage. Attackers may use NFVI or hypervisor vulnerabilities to their advantage. For instance, an attacker might be able to compromise confidentiality and integrity.

In addition to exploiting the availability of VNF resources, attackers may attempt to escape from the virtual computing, network, or storage environment in order to gain access to the host's physical compute, network, or storage resources. A secure boot system that offers some type of integrity protection can be enabled as a solution to this problem, giving users confidence that the hypervisor has not been tampered with.

VNF and Application layer

Relate to the virtualization of network functions. In NFV systems, the entire framework is supported by a network graph that combines all of the software packages that represent the various network services. NFs could be the source of an assault or its victim. As a VNF is a vendor-provided component that is generally independent of the infrastructure provider, it may include security holes or even malicious software that is intended to launch attacks. In order to provide secure management, this layer solutions must have appropriate identity and access management as well as certificate management. Application-specific best practices for application security should enable automated security policy, VNF scanning, and continuous security monitoring.

Spectre Attacks

There are two ways to provoke and sway incorrect speculative execution.

Exploiting Conditional Branches (Variant 1(V1))

During a Spectre attack of this nature, the attacker trains the CPU's branch predictor to follow an incorrect path, thereby enabling the CPU to execute code that should not have been executed, and temporarily breaking the program's semantics. As a result of this improper speculative execution, an intruder can gain access to confidential data stored in the program's primary memory. Consider the below subcode as an illustration:

```
if z < ary1_size
print("x=ary2[ary1 [z] *4096]")
```

Consider that the parameter z in the previous example has information that the attacker has full control over. The aforementioned procedure contain an if statement, the aim of that to ensure *z* value that an acceptable domain and thus to get guaranteed legitimacy of the main memory of *ary1*. Bypassing this if statement, an attacker potentially obtains possibly confidential information based on the process' address space.

An attacker initially runs the aforementioned code with genuine arguments even during preliminary mistraining step, teaching the predictor branch to foresee that the condition that would be right. The attacker then executes the code during the exploit step with a value of *z* outside of the boundaries of *ary1*. The CPU starts speculatively executing instructions that compute ary2[ary1[z]*4096] that used the erroneous z before knowing the branch outcome because it expects that the bounds check will be successful. Be aware that the call from ary2 puts data into the cache at a location which depends on *ary1[z]* using the malicious *z*, scaled to ensure that authenticates travel to separate cache lines and prevent the impacts of equipment prefetching.

Once the results of the boundary check are obtained, the CPU detects its mistake and reverts any modifications made to its original microarchitectural state. Nevertheless, modifications to the cache state cannot be reversed, and an intruder can examine the cache contents to determine the value of any potentially sensitive bytes obtained through an out-of-bounds read from the victim's memory.

Exploiting Indirect Branches (Variant 2 (V2))

In this variation, which is based on return-oriented programming (ROP).[20,21,22] In this attack scenario, the attacker chooses a device from the victim's address space and convinces the victim to hypothetically execute it. Unlike in a ROP attack, the attacker does not rely on a vulnerability in the victim's code. Instead, the attacker manipulates the Branch Target Buffer (BTB) to predict a branch from an indirect branch instruction to the gadget's location, leading to the gadget being executed speculatively. As previously, erroneous speculative execution has an effect on the nominal state of the CPU but has no effect on the cache, allowing the device to leak private data through a cache side channel.

The attacker locates the gadget's virtual address in the address space of victim's, [23,24] subsequently conducts indirect branches for mentioned address in order to confuse the BTB. The address space of the attacker is used for this training. It is not important what is located All that is required during a training session is for the virtual addresses of the attacker and victim to coincide at the gadget address in the attacker's address space.

In reality, The attack can be successful even if the attacker's address space does not contain any code assigned to the device's virtual address, provided that the attacker can handle exceptions appropriately.

Changing the methods for obtaining speculative execution and unauthorized disclosure can result in the creation of new attacks. For instance, by improperly training return instructions, exposing timing fluctuations that reveal information, or creating conflicts on arithmetic units,[25] attackers can develop new methods of attack.

METHODS

In this section we will show how we directed the attacks on the architecture proposed in below figure (figure 2).

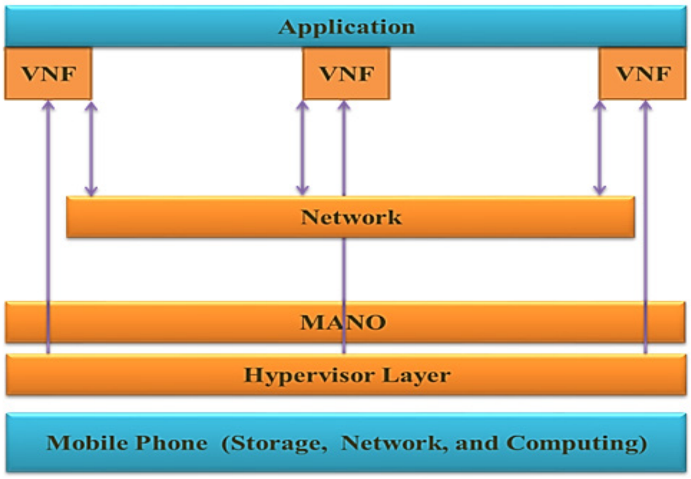


Figure 2. Suggested architecture

Attacks of the Hardware

In this aspect, we generated a specter attack on a mobile that contains the system (figure 3), based on execution of speculative and prediction of a branch. After that, we could run code on the intended system.

Take into account the scenario where the following code (conditional branch) is a component of a function that receives a dubious source's unsigned integer z. The running the code can access There are two arrays: the first one is called ary1 and has a size of "ary1 size", while the second one is called ary2 and has a size of 2 MB.

```
if z < ary1_size
print("x= ary2[ary1 [z] *4096]")
```

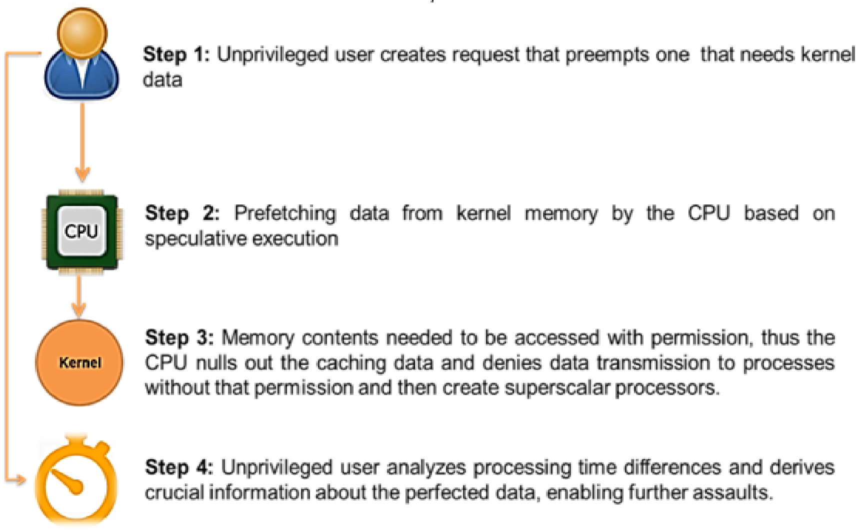
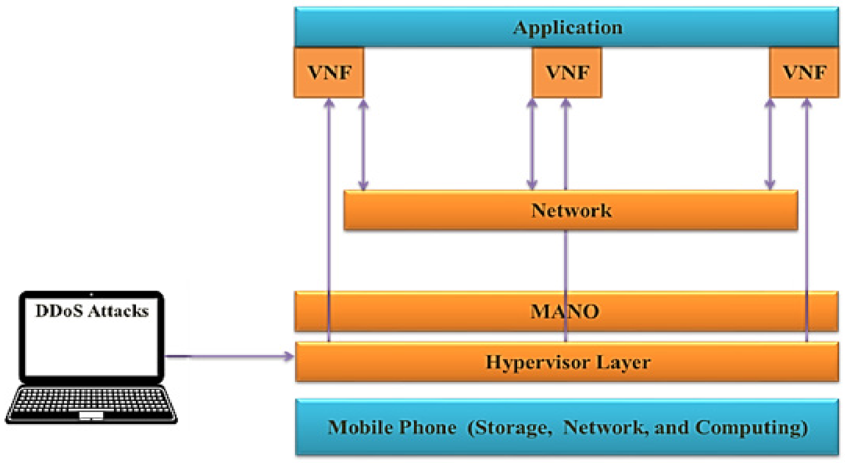

Step 1: Unprivileged user creates request that preempts one that needs kernel data
CPU
Step 2: Prefetching data from kernel memory by the CPU based on speculative execution
Kernel
Step 3: Memory contents needed to be accessed with permission, thus the CPU nulls out the caching data and denies data transmission to processes without that permission and then create superscalar processors.
Step 4: Unprivileged user analyzes processing time differences and derives crucial information about the perfected data, enabling further assaults.

Figure 3. Specter attack generation

Hypervisor Attack (DoS)

An attempt to take over the hypervisor that oversees the virtual environment is known as a hyperjacking attack. This is feasible if an attacker inserts a hypervisor directly below the existing one or gains access to the existing hypervisor. We generated a hypervisor DoS (in below steps) as a test attack to the hypervisor in the proposed architecture via a virtual machine's rootkit installation (figure 4).



.

Figure 4. Denial-of-Service Generation

Now, we listed many steps for hypervisor DDoS generation based on many servers:

- Firstly, we will have to deploy Mc-word file on a virtual device as we'll use it as an example program and hijack its updating.
- Then, install evilgrade, a modular gap that enables the attacker to profit from subpar upgrade performances by sneaking "bad" upgrades past an unsuspecting victim.
- Then, execute Evilgrade.
- Then, hijacking the upgrade of MS-word file.
- Then, describe the program we intend to use to hijacking evilgrade's upgrading process.
- Then, establish the conditions under which the evilgrade MS-word module must operate.
- Then, with the use of Metasploit, create a malicious payload to be sent to the software upgrader in place of the genuine update.
- Then, run evilgrade server.
- Then, include MS-word Upgrade Server in the list of domains to hijack.
- Then, via a MitM attack, route the traffic via the network by Configure Iptables.
- Then, run Ettercap to examine network data going over a computer interface.
- Then, run Netcat Listener.

- Then, open Word Documents on the Windows victim computer.
- Finally, check for DNS Spoof.

RESULTS AND DISCUSSIONS

Regarding the lower layer and the results that appeared after the two experiments, which we mean before and after the attacks The result of the hardware device was:

- Accessing the root of the system.
- Access to the memory of a target or victim process executing on the system.
- Furthermore, the attack is executed in a scenario affecting a virtualized medium.

Then the solutions we add it to the architecture to prevent or reduce the impact of these attacks are:

- Software secures e.g. browsers. Bios (EFS filesystem, bootloader) or KNOX version updates as shown in figure 5.

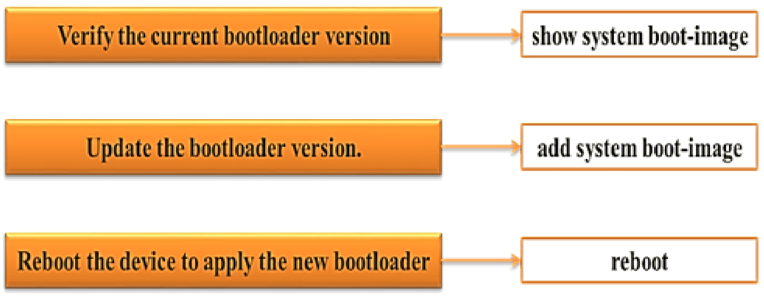

Verify the current bootloader version
show system boot-image
Update the bootloader version.
add system boot-image
Reboot the device to apply the new bootloader
reboot

Figure 5. Bootloader Update

Table 1 shows the outcomes for the detection of Spectre attack before solutions.

Table 1. Detection of Spectre attack before solutions

Program	Size	Time of analysis (h&m)	Conditional branches
WinRAR	1,4MB	126h	21,141
NCH Prism	3,1MB	150h	58,522
hmmer	317KB	3,40h	4,211
IBM Blockchain Platform	4MB	166h	63,214
Kaleido	50KB	21m	538

There is a clear difference in the table 2, which was obtained and analysed for its time after the solutions that were developed and gave protection to the hardware device.

Table 2. Detection of Spectre attack before solutions

Program	Size	Time of analysis (h&m)	Conditional branches
WinRAR	1,4MB	128h	22,141
NCH Prism	3,1MB	151h	59,522
hmmer	317KB	3,45h	4,255
IBM Blockchain Platform	4MB	169h	63,314
Kaleido	50KB	22m	540

Also, in the figures 6 and 7 we notice a clear difference in the tainted conditional branches, Specter Variant 1 (V1) condition and Specter Variant 2 (V2), which were tested before and after the update. All three are clearly increased by increasing the security of the proposed architecture.

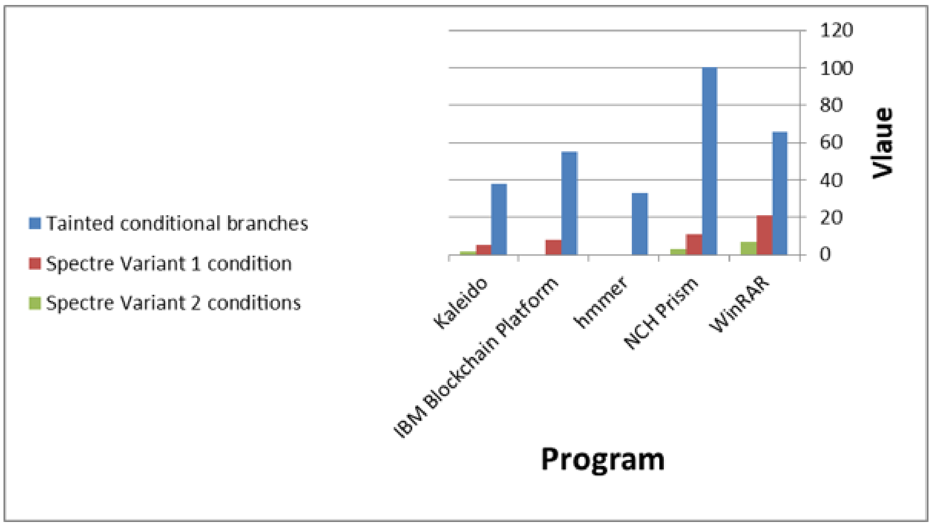


Figure 6. Comparison between TB, Spectre V1 and Spectre V2 before solutions

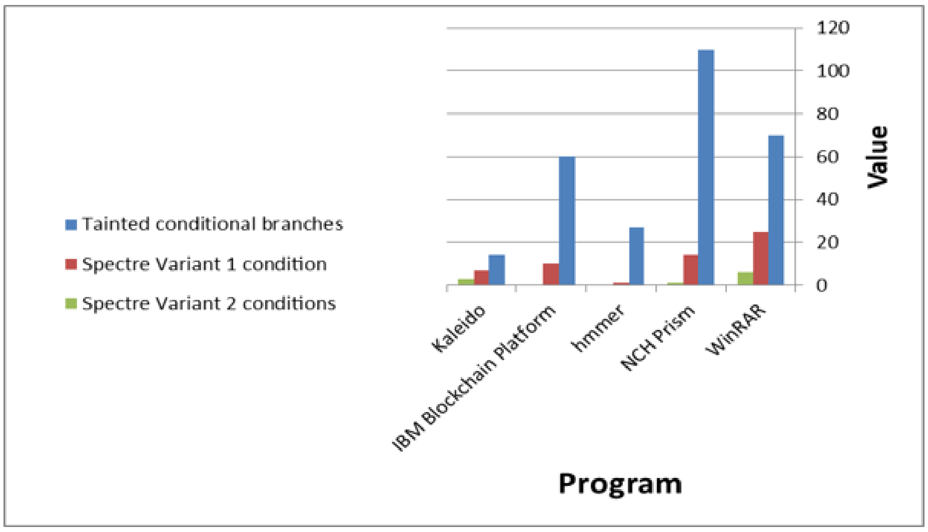


Figure 7. Comparison between TB, Spectre V1 and Spectre V2 after solutions

With regard to the important layer the hypervisor layer through which it is possible to fully operate on the network in addition to its services. Each hack that occurs at the hypervisor level has a major detrimental impact, as it grants attackers complete control over the virtualized environment and enables them to conceal their malicious actions. We did several procedures through which the attacks were monitored or gave a great opportunity to the network administrator to secure the architecture. As shown in the table 3 and figure 8, below refer the network before the updated the hypervisor and execute procedures.

Table 3. Time of attacks before update the hypervisor

Attacks	Time to attack (hours)	Alert
Denial of Service	112	No
Non-updated Hypervisor	123	No

The first procedure, as soon as the vendor provides an update, install it on the hypervisor. Most hypervisors include capabilities that will automatically look for modifications and install them if any are detected. In addition to that sending an alternation for manager about the attacks. As shown in below table 4 and figure 9, that indicated to the attacks time after hypervisor updated.

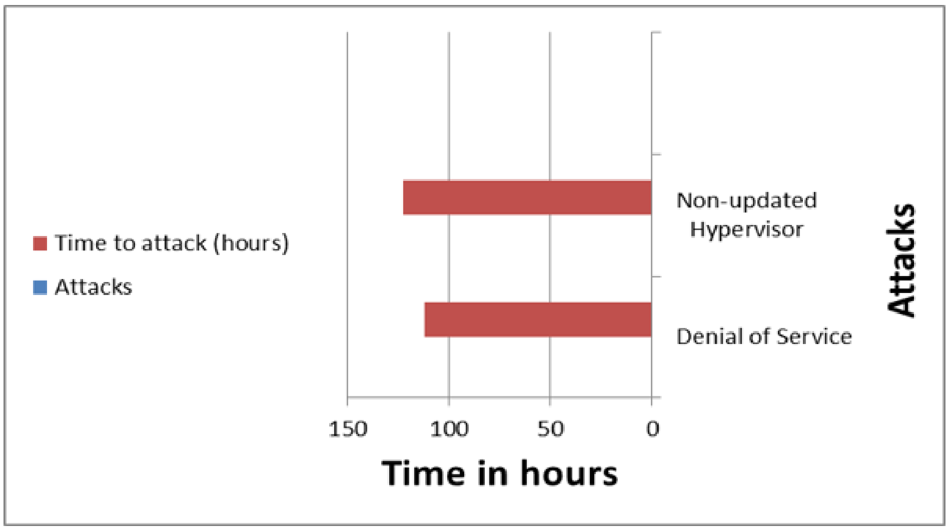


Figure 8. Time of attacks before update the hypervisor

Table 4. Time of attacks after update the hypervisor

Update	**Time of attack (hours)**	**Alert**
vCenter/ESX	188	Yes
Hyper-V	176	Yes

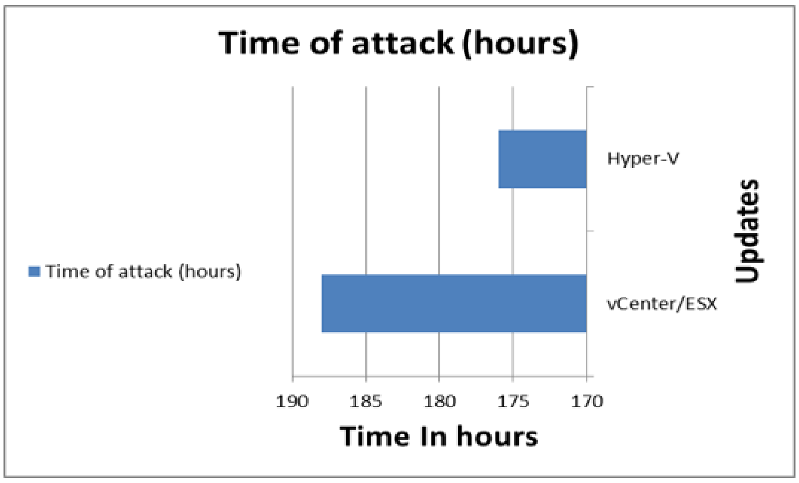


Figure 9. Time to attacks after updates the hypervisor

The second procedure, if they are not needed, it is recommended to disable all hypervisor services such as file and clipboard sharing between the guest and host operating systems.

The last procedure, utilize introspection tools to keep an eye on the privacy of interactions among host kernels.

CONCLUSION

This paper discusses the many fields of IT, Firstly the work of NFV in the smartphone and how employed with complex services of these mobiles. Secondly, the security field of NFV tools and smartphones used attacks like Specter and DoS. We are finally merging these two fields to secure the virtual smartphone based on NFV. We obtained different time before/ after hypervisor layer updated that indicates the effectiveness and strength of the updates used.

REFERENCES

1. Han B, Gopalakrishnan V, Ji L, Lee SJ. Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine. 2015;53(2):90-97.

2. Cotroneo D, De Simone L, Iannillo AK, Lanzaro A, Natella R, Fan J, et al. Network function virtualization: Challenges and directions for reliability assurance. In: Editor (Ed.)^(Eds.). Book Network function virtualization: Challenges and directions for reliability assurance. IEEE; 2014. p. 37-42.

3. Jawdhari HA, Abdullah AA. The application of network Functions Virtualization on different networks, and its new applications in blockchain: A survey. 2021.

4. Virtualisation NF. Introductory white paper. Editor (Ed.)^(Eds.). 2012. p.

5. Hawilo H, Shami A, Mirahmadi M, Asal RJ. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). Network. 2014;28(6):18-26.

6. Gember-Jacobson A, Viswanathan R, Prakash C, Grandl R, Khalid J, Das S, et al. OpenNF: Enabling innovation in network function control. ACM SIGCOMM Computer Communication Review. 2014;44(4):163-174.

7. Soares J, Gonçalves C, Parreira B, Tavares P, Carapinha J, Barraca JP, et al. Toward a telco cloud environment for service functions. IEEE Communications Magazine. 2015;53(2):98-106.

8. Ding W, Qi W, Wang J, Chen B. OpenSCaaS: an open service chain as a service platform toward the integration of SDN and NFV. Network. 2015;29(3):30-35.

9. Montero D, Yannuzzi M, Shaw A, Jacquin L, Pastor A, Serral-Gracia R, et al. Virtualized security at the network edge: A user-centric approach. Network. 2015;53(4):176-186.

10. Lal S, Taleb T, Dutta A. NFV: Security threats and best practices. IEEE Communications Magazine. 2017;55(8):211-217.

11. ETSI Group Specification. Network Functions Virtualization (NFV) NFV Security and TrustGuidance. Dec. 2014.

12. Dulphy J-P, Demarquilly C, Boisseau J, Bousquet H, Lhotelier L. Etude du comportement alimentaire et mérycique de moutons recevant des fourrages verts hachés. Editor (Ed.)^(Eds.). Book Etude du comportement alimentaire et mérycique de moutons recevant des fourrages verts hachés. 1974. p. 193-212.

13. Arya N, Gidwani M, Gupta SK. Hypervisor security-A major concern. International Journal of Innovations in Engineering and Technology. 2013;3(6):533-538.

14. Szefer J, Keller E, Lee RB, Rexford J. Eliminating the hypervisor attack surface for a more secure cloud. Editor (Ed.)^(Eds.). Book Eliminating the hypervisor attack surface for a more secure cloud. 2011. p. 401-412.

15. Jawdhari HA, Abdullah AA. A novel blockchain architecture based on network functions virtualization (NFV) with auto smart contracts. IEEE Open Journal of the Communications Society. 2021;9(4):834-844.

16. Haleplidis E, Pentikousis K, Denazis S, Salim JH, Meyer D, Koufopavlou O. Software-defined networking (SDN): Layers and architecture terminology. Editor (Ed.)^(Eds.). Book Software-defined networking (SDN): Layers and architecture terminology. 2015.

17. Alwakeel AM, Alnaim AK, Fernandez EB. A Pattern for a Virtual Network Function (VNF). Editor (Ed.)^(Eds.). Book A Pattern for a Virtual Network Function (VNF). 2019. p. 1-7.

18. Liang C, Yu FR. Wireless network virtualization: A survey, some research issues and challenges. IEEE Communications Surveys & Tutorials. 2014;17(1):358-380.

19. 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 Communications Surveys & Tutorials. 2015;18(1):236-262.

20. Shacham H. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). Editor (Ed.)^(Eds.). Book The geometry of innocent flesh on the bone: Return-into-libc without function

calls (on the x86). 2007. p. 552-561.

21. Cloosters T, Paaßen D, Wang J, Draissi O, Jauernig P, Stapf E, et al. RiscyROP: Automated Return-Oriented Programming Attacks on RISC-V and ARM64. Editor (Ed.)^(Eds.). Book RiscyROP: Automated Return-Oriented Programming Attacks on RISC-V and ARM64. 2022. p. 30-42.

22. Shrivastava RK, Singh SP, Hasan MK, Islam S, Abdullah S, Aman AHM. Securing Internet of Things devices against code tampering attacks using Return Oriented Programming. Computers & Electrical Engineering. 2022;193:38-46.

23. Tobah Y, Kwong A, Kang I, Genkin D, Shin KG. SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks. Editor (Ed.)^(Eds.). Book SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks. 2022.

24. Roney J, Appel T, Pinisetti P, Mickens J. Identifying Valuable Pointers in Heap Data. Editor (Ed.)^(Eds.). Book Identifying Valuable Pointers in Heap Data. IEEE; 2021. p. 373-382.

25. Canella C. Hardening the Kernel Against Unprivileged Attacks. 2022.

FUNDING
No financing.

CONFLICTS OF INTEREST
None.

AUTHOR CONTRIBUTIONS
Conceptualization: Hayder A. Nahi, Mustafa Asaad Hasan, Ali Hussein Lazem, Mohammed Ayad Alkhafaji.
Investigation: Hayder A. Nahi, Mustafa Asaad Hasan, Ali Hussein Lazem, Mohammed Ayad Alkhafaji.
Methodology: Hayder A. Nahi, Mustafa Asaad Hasan, Ali Hussein Lazem, Mohammed Ayad Alkhafaji.
Writing - original draft: Hayder A. Nahi, Mustafa Asaad Hasan, Ali Hussein Lazem, Mohammed Ayad Alkhafaji.
Writing - review and editing: Hayder A. Nahi, Mustafa Asaad Hasan, Ali Hussein Lazem, Mohammed Ayad Alkhafaji.