World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:16, No:6, 2022

Impact of Network Workload between Virtualization Solutions on a Testbed Environment for Cybersecurity Learning

Kévin Fernagut, Olivier Flauzac, Erick M. Gallegos R, Florent Nolot

Abstract—The adoption of modern lightweight virtualization often comes with new threats and network vulnerabilities. This paper seeks to assess this with a different approach studying the behavior of a testbed built with tools such as Kernel-based Virtual Machine (KVM), LinuX Containers (LXC) and Docker, by performing stress tests within a platform where students experiment simultaneously with cyber-attacks, and thus observe the impact on the campus network and also find the best solution for cyber-security learning. Interesting outcomes can be found in the literature comparing these technologies. It is, however, difficult to find results of the effects on the global network where experiments are carried out. Our work shows that other physical hosts and the faculty network were impacted while performing these trials. The problems found are discussed, as well as security solutions and the adoption of new network policies.

Keywords—Containerization, containers, cyber-security, cyber-attacks, isolation, performance, security, virtualization, virtual machines.

I. Introduction

THE great success of classic virtualization has decreased in recent years due to the growing limitations and deficiencies in performance and resource management on a large scale. Despite being a milestone during the last decades offering a high degree of security reducing network impact, costs and improving user experience; researchers and computer experts are leaning towards new lightweight solutions such as containers and Unikernel-type systems in order to reduce heavy virtualization overhead. However, containers are known to be less secure than virtual machines (VMs), and the provision of cloud computing services where millions of users are given access, requires an approach that is imperatively oriented towards security, isolation and performance. The majority of studies in the literature focus on analysing virtualization solutions in fixed environments, but not the impact these can have on the network in which they are being tested while someone else is running tests on other machines. Workload tests carried out on a shared environment can have an important impact not only on the interconnections of VMs [1], the testbed itself or the technology used, but also on other physical equipment and the global network. Therefore, in this paper, we carried out stress tests on a virtualization platform called Remotelabz, used by students at the University to find the most suitable solution for teaching cyber-security by comparing and determining the limits of VMs and containers put under pressure, as well as observing the impact on the campus network, finding possible problems and allowing the enhancement and implementation of new policies within the IT department for an access to more secure physical and virtual environments. Hence in Section II, we present the state of the art and the related work along with a description of existing virtualization solutions technologies and tools. In Section III, our experimental testbed and protocol is described. In Section IV. the details and results of a first set of stress tests are developed. Section V presents multiple stress tests performed in a scenario closer to reality and their results. In Section VI, the problems found during testing are discussed and different solutions are proposed. In Section VII, we present our conclusions and perspectives for future works.

Kévin Fernagut, Olivier Flauzac, Erick M. Gallegos R, and Florent Nolot are with CReSTIC laboratory, EA 3804, 51097, at the University of Reims Champagne-Ardenne, Moulin de la Housse - BP 1039 - 51687 Reims France (e-mail: olivier.flauzac@univ-reims.fr, florent.nolot@univ-reims.fr, erik.gallegos-robledo@univ-reims.fr, fernagut.kevin@etudiant.univ-reims.fr).

II. State of the Art

With the growing number of users accessing services online, big companies found themselves in the need to make sure their infrastructure was capable and scalable enough to provide uninterrupted availability. To achieve this, they relied on hardware and resources that were becoming expensive to maintain for a single service. Semnanian et al. [2] mention in their work how hardware was built on the basis of running a single application, squandering most of the resources available. This led to the use of virtualization technologies.

A. Virtualization

Virtualization is a widely used technology in a lot of domains such as education, high performance computing (HPC) [3], research and cloud computing services by using an abstraction of processes or a complete system known as VMs [4]. These can switch from one device to another, eliminating the concern that, in the event of a device failure, a network or a system can continue to function. The main objective is to facilitate the construction of new systems without the constraints of traditional hardware and software, enhancing performance, reducing network impact, costs and providing isolation with different methods and levels of security, making them less prone to events that may make them irreplaceable. In other words, it simplifies complex systems by maintaining their essence and functionality, transforming a physical object into an ideal abstract object. However, its complexity goes beyond depending on the level of abstraction. In [5], Smith et al. describe the taxonomy of VMs and unify their different

concepts and architectures. Understanding how virtualization works, can help reduce considerably the complexity and help define better interfaces. Roy et al. [6] point out the need for accurate emulation and that working with fixed network resources on a controlled testbed can limit the tests, and impact the accuracy of results, hiding potential problems that can only be found on a large scale.

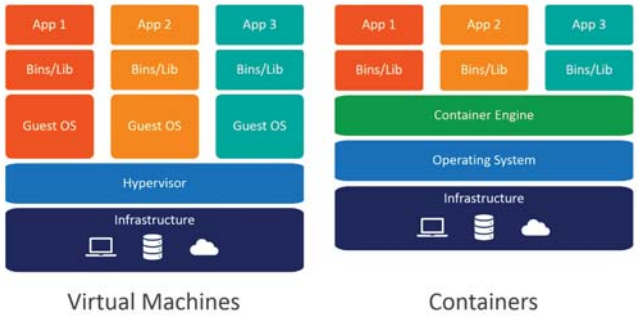

Fig. 1 Difference between virtual machines and containers

B. Containerization

Despite being more secure as a result of their complete isolation the main problem with classic VMs is the resources required to emulate software and hardware which are in turn independent on real hardware capabilities, so the number of instances on a host are limited to the resources available. This is where containers come into play, which are close to VMs, but instead they do not emulate hardware or use a complete OS inside of a container. Pahl et al. [7] make an extensive review and taxonomically describe, classify and compare containers and their orchestration. Contrary to VMs (Fig. 1), a container uses multiple isolated processes on the host's OS instead of installing one for each container, reducing overhead by encapsulating applications and dependencies. The main purpose is to isolate applications with different methods and levels of security to perform specific tasks limiting the use of resources and preventing the impact on the main system and the network where the container is hosted. Simply put, they can allow changes to the applications they hold without affecting their surrounding environment. In [8], the authors compare multilevel virtualization impact and performance in cloud computing, but they conclude that using different levels of virtualization configurations performance did not decrease significantly. In contrast, Joy [9] compares the performance and scalability of VMs against containers using macro benchmarks showing that containers do a better job than VMs, but when it comes to handling critical business data the latter are better. However this work only shows the impact on the testbed itself and not the surrounding infrastructure. Nevertheless, this exposes the downside with containers on their dependency on a normal OS and the fact that a non-secure container with root privileges can be an access point in an insecure infrastructure, by privilege escalation, or DOS attacks to saturate a container and consequently the host and the network itself. The degree of elasticity that can make virtualization easier to adapt to what is needed and provide resources quickly and efficiently resides on the right choice of tools described in Subsection *C*.

C. Virtualization Tools

Building a proper secure testbed, which is an environment in which we can test, develop and play with different features, generally combines different technologies, making them fast and precise depending on what is being used. In [10], it is concluded that some technologies performed better depending on what is the purpose and what is being observed on the system. Testbeds can be shared, but they are usually difficult to configure if the user does not know the proper technology to be used. In [11], Huang et al. review different technologies and cover the requirements of network testbeds describing different possible attacks and the importance of security no matter what modern virtualization tool is introduced, still being vulnerable to security attacks.

Among the wide range of virtualization and containerization tools existing out there, these can be regrouped into different categories: simulators, emulators, hypervisors, containers and unikernels. A simulator as its name suggests, only simulates

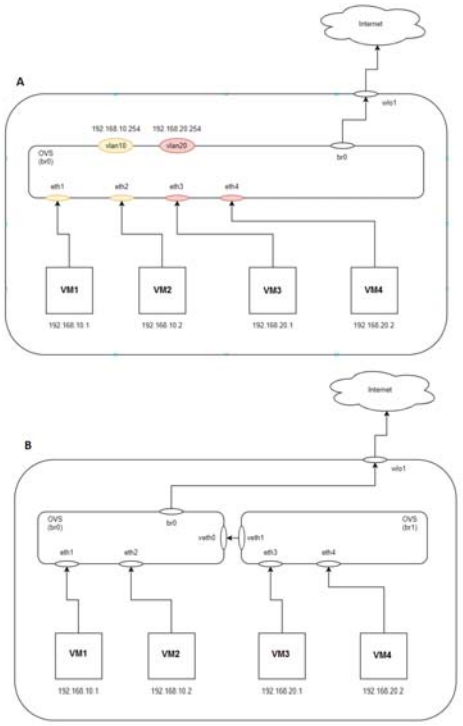

Fig. 2 (A) Single OVS with VLANs; (B) Dual OVS without VLANs

the behavior and functions of an object. This makes its features useless if real behavior is needed, often requiring code modification and usually known to be slow. Some examples of simulators are GNS3/Dynamips/Dynagen, Cisco Packet Tracer, NetSim, QualNet. In [12], Siraj et al. study the main features, advantages and disadvantages of these tools. Emulators are an exact copy of real hardware or software but depend largely on hardware resources. Emulators examples are Open vSwitch (OvS), OpenDaylight, CORE, NetEm, EVE-NG, Cisco VIRL. In [13], Kondratyuk presents a

comparison between simulators and emulators. Hypervisors or monitors are programs that allow translations between physical and virtual resources. There are two types [14]. Type 1, also called bare-metal, have direct access to real hardware resources, they are more secure and security flaws and vulnerabilities are reduced because they do not depend on a vulnerable operating system. Examples are VMware vSphere, Microsoft Hyper-V, Kernel Virtual Machine (KVM) or Qemu with KVM support. Type 2, known as hosted hypervisors, depend on an operating system to manage resources, which causes what is called VM network latency because it handles not only what is happening in the virtual environment, but also what happens in the host's operating system. Examples are VMware Fusion, Oracle VirtualBox, Oracle Solaris Zones, VMware Workstation and Qemu without KVM support.

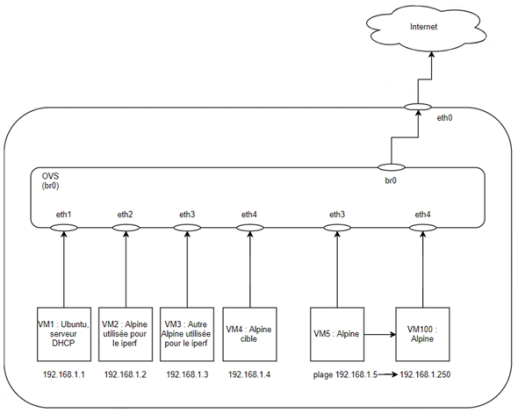

Fig. 3 Single OVS with 100 containers

Regarding containers there is a variety of tools that use this technology. Examples are LXC, Docker, Singularity, OpenVZ. Each platform provides different features, but the containers operates on the same principle. Even though Unikernels will be discussed in a future work. It is important to note their features which, as opposed to containers, they compile a custom OS with the minimal set of libraries that include only the functionality required by the application. Some examples are MiragrOS, Rump Kernels, ClickOS, cLIVE, IncludeOS. In [15], Kurek compares these three technologies on a testbed consisting of bare metal nodes, but the results focuses only on the interaction between the nodes.

Knowing the different tools available for virtualization, we continue our work by describing the environment and the elements we chose for our study.

III. Experiment Settings

This section presents the environment built for the experimentation allowing the study of the impact on servers and infrastructure hosting the RemoteLabz platform [16] while performing two sets of stress tests on a testbed sharing the same infrastructure. RemoteLabz is a tool based on VMs and containers set up by the University to perform remote work in the IT field, allowing students and users access to dedicated virtual environments via a simple web browser. The main objective is to orient this tool towards the field of cyber-security and be used as a cyber-range tool, observing the risks and evaluate its resistance under a cyber-attack by generating big amounts of traffic that might have a significant impact on the campus network if any deficiencies are found.

A. Hardware

Our experimental testbed was built using four physical hosts:

- Host n1: An Intel i5-3230 4 core processor at 2.6GHz and 4 GB of RAM with Ubuntu 16.04.
- Host n2: An Intel i7-6700 8 core processor at 3.4GHz with 32 GB of RAM with Ubuntu 18.04.
- Host n3: An Intel i7-6700 8 core processor at 3.4GHz with 32 GB of RAM with Ubuntu 18.04.
- Host n4: A Xeon E5-2630L processor at 1.8GHz with 32 GB RAM with Debian 4.15.11-1.on VMware [17] (this will allow us to configure processor cores at will).

B. Virtual Environment

Our testbed is divided into four architectures. In the first architecture A (Fig. 2) four VMs were set up, interconnecting

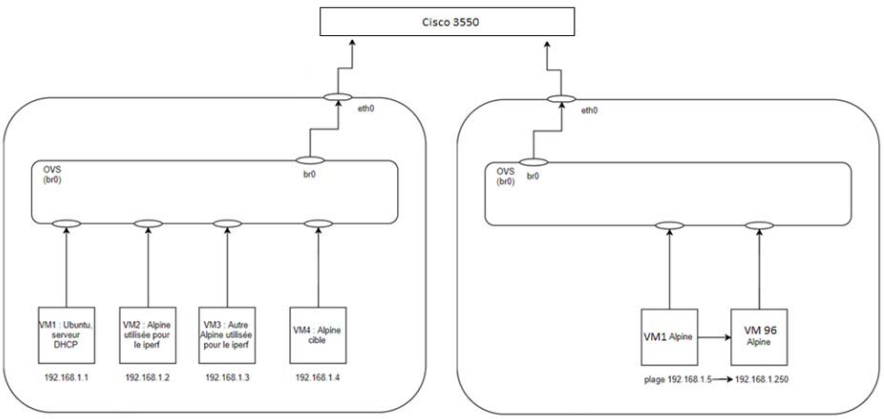

Fig. 4 Double OVS with 100 VMs/Containers

TABLE I
KVM PERFORMANCE TEST SUMMARY

Host Config	Intel Core i5-3230 2.6GHz 4 core 4Go RAM		Intel Core i7-6700 3.4GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 2 core 32Go RAM	
OVS Config	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN
Condition	Normal							
Bandwidth between VM1 and VM2 (Gb/s)	0.933	0.935	3.8	3.9	0.840	0.871	0.716	0.773
Condition	hping3 –flood –rand-source -d 200 -S between VM3 and VM4							
# packets iperf	100 000		460 000		90 000		90 000	
Bandwidth between VM1 and VM2 (Gb/s)	0.345	0.396	2.89	2.94	0.792	0.871	0.373	0.330
Bandwidth loss %	62.94	57.67	24.32	25.71	5.69	0,057	47.91	57.27

them using a virtual switch (vswitch), two of them in VLAN 10 and two others in VLAN 20. Two gateway interfaces were as well put in place for the inter-VLAN routing. Then in a second architecture B (Fig. 2), the same VMs were set up, but this time distributed on two different vswitches without VLANs. These architectures will be used for a first set of stress tests. For the second set of stress tests, a third architecture (Fig. 3) was set up with 100 VMs/containers with a simple vswitch. A first container will serve as a local DHCP server. Two containers are left intact in order to run bandwidth measurements. A fourth container is the target by running denial of service attacks simultaneously on the remaining 96 containers. The fourth architecture (Fig. 4). was divided into two parts, set up on two separate physical hosts interconnected using a cisco 3550 physical switch and a vswitch installed on each. The same principle is used as in (Fig. 3) but this time the first four VMs/containers will be on the first host, and the other 96 on the second one.

C. Software

The main goal of the experimentation is to carry out systematic network cyber-attacks, therefore hping [18] was chosen, a tool that allows multiple pings while modifying IP packets making custom size TCP, UDP requests on a specific port. It also allows to perform denial of service attacks, by sending thousands of packets per second. It is thus this option that will try to saturate the virtualized testbed. Along with hping, Iperf [19] was used, which is a server-client tool to perform bandwidth measurements on TCP flow generated between hosts.

In order to carry out the testing without the need of a physical switch, OpenvSwitch (OvS) [20] was used to set up a vswitch and separate VMs using VLANs. OVS is an open source project that allows the virtualization of a physical switch. It works like any physical Layer 2 and Layer 3 switches, with the advantage that features can be added and removed, such as the limitation in the number of ports.

For the VMs and containers, KVM, LXC and Docker were used. This allows the collection of all the necessary data to compare the different results. KVM [21] is a linux based hypervisor that allows the creation of VMs as linux isolated processes integrating with the rest of the system. LXC [22] the linux container solution, is the abbreviation of ”LinuX Containers”. It is a widely used solution included in the linux kernel since the 2.6.24 version. It works along with cgroups and namespaces. To observe the behavior of the testbeds on a more robust containers solution Docker [23] was used, which is an extension of LXC adding more functionalities to manage data, processes and isolation. It provides a Docker Hub, which is a container library with over 100000 container images that works as a server-client application.

IV. STRESS TESTING DESCRIPTION

In this section, the first set of stress tests is explained in detail in order to see the resistance to attacks on the architectures A and B, proceeding first with KVM VMs, LXC containers and then Docker containers, concluding with an assessment of what has been achieved and observed in this part.

A. KVM VMs Test

In our first approach four KVM VMs were used on the physical host n1, performing bandwidth measurements with iperf on architecture A, first in normal time by running the test 10 times between VM1 and VM2 obtaining an average bandwidth of 933.3 Mbps. Then launching hping along with iperf between on VMs 3 and 4 to try to saturate the vswitch and observe its reaction, giving an average bandwidth of 345.8 Mbps. This represented a loss of 62%, which is a noteworthy difference leading to the assumption that the vswitch could have been either saturated with a simple hping, or the VLANs did not isolate well. To dispel this concern, a second test was carried out on architecture B with the distributed VMs on the two OvS’, attaining a similar loss of 57% of bandwidth. This introduced the possibility that such loss did not come from the host having only 4 GB of RAM, but rather an old processor slowing down the execution of the VMs or the OvS.

To get a clearer view of this issue host n2 was used having a more powerful processor. With the new configuration, an

TABLE II
LXC PERFORMANCE TEST SUMMARY

Host Config	Intel Core i5-3230 2.6GHz 4 core 4Go RAM		Intel Core i7-6700 3.4GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 2 core 32Go RAM	
OVS Config	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN
Condition	Normal							
Bandwidth between VM1 and VM2 (Gb/s)	30.39	31	67.02	67.24	26.16	26.51	31.89	26.74
Condition	hping3 –flood –rand-source -d 200 -S between VM3 and VM4							
# packets iperf	755 000		1 590 000		480 000		480 000	
Bandwidth between VM1 and VM2 (Gb/s)	21.69	23.39	65.13	66.57	24.47	26.57	21.58	18.94
Bandwidth loss %	28.63	24.55	2.82	0.357	6.4	0	32.33	29.17

average bandwidth of 3.828 Gbps was observed with a test in normal time on architecture A, and an average of 2.897 Gbit/s by running hping between the two other VMs on the second test, getting a 24.32% loss of bandwidth. The testing continued with architecture B, with an average bandwidth of 3.963 Gbps in normal execution. Next with hping between two VMs on the other vswitch, resulting in an average bandwidth of 2.944 Gbit/s, giving a loss of 25.71%.

As shown in the first two hosts, the results obtained on both architectures A and B are relatively similar giving a clear answer that the processor was indeed important. On host n2, the bandwidth loss went down to only 25%, an increase in bandwidth important to consider with the number of packets sent by the hping, which was much more important on this host with 46000 packets/s against 8000 packets/s on host n1. The same tests were performed as before but this time using host n4 because the number of cores on a VM can be easily modified when performing tests on a VMware platform, therefore the new tests were done putting 8 cores on the VM. On a VM set up with 8 cores architecture A was hardly impacted during testing with an average of 840.1 Mbit/s in normal time and an average of 792.3 Mbit/s using hping, obtaining a 5.69% bandwidth loss. By scaling down to two cores the results were as expected with an average of 716.4 Mbit/s in normal time and 373.2 Mbit/s with hping, having a loss of 47.91% of bandwidth. In both configurations the number of packets sent by hping were similar. This shows us the importance of the processor to avoid the saturation and slowing down the execution of the vswitch and the traffic. The results obtained on architecture B were almost similar. The summary of all the results of these tests are shown in Table I.

B. LXC Containers Test

After testing KVM, LXC containers were used with the same tests on host 1. On architecture A and B, the results obtained on the first two tests were quite similar, 30.39 Gbit/s and 31Gbit/s bandwidth respectively in normal time. The first thing noticed with the containers is that speed is much higher ranging from a few Mbit/s to several Gbit/s. Then, during the hping testing, the bandwidth resulted with 21.69 Gbit/s for architecture A, and 23.39 Gbit/s for architecture B, losing 28.63% and 24.55% respectively. The results were better than those obtained with KVM VMs under the same tests, less than 30% versus more than 60% with KVM. However, the loss was higher than expected. This could be explained by looking at the number of packets sent by hping being more than 75000/s against barely 10000/s with KVM.

Next we move over to host 2. The results were very close regardless of the architecture, with a bandwidth in normal time of around 67 Gbit/s. The power of the host being higher, the hping sent a total of 159,000 packets/s. A significant loss was expected, however during the hping on architecture A, a bandwidth loss of about 2.82% was observed and 0.357% for architecture B. The CPU was not saturated, so the results confirm that only the CPU was causing losses. This time the VMWare platform was used on a host n4. This resulted in very few losses just like host n2, but with a lower bandwidth, 26 Gbit/s vs 67 Gbit/s on host 2. After lowering the number of cores down to 2, we went back up to a loss of more than 30%.The summary of all the results of these tests with LXC are shown in Table II.

C. Docker Containers Test

In order to compare the LXC results with another container solution, we used Docker. However, the results did not change, apart from a few values, very similar to those obtained with LXC. The summary of all the results of the tests done with Docker can be seen in Table III.

V. MULTIPLE STRESS TESTING DESCRIPTION

In this section we decided to perform multiple stress tests in scenarios closer to reality with 100 VMs and containers under the second set of architectures 3 and 4.

A. LXC Containers Test

Scenario 1 architecture (Fig. 3) on host n2: The results obtained by performing a test in normal time using iperf between container 2 and container 3 gave an average bandwidth of 64.93 Gbit/s. Then by executing hping on the

TABLE III
DOCKER PERFORMANCE TEST SUMMARY

Host Config	Intel Core i5-3230 2.6GHz 4 core 4Go RAM		Intel Core i7-6700 3.4GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 8 core 32Go RAM		Intel Xeon E5-2630L 1.8GHz 2 core 32Go RAM	
OVS Config	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN	1 OVS & VLANs	2 OVS w/o VLAN
Condition	Normal							
Bandwidth between VM1 and VM2 (Gb/s)	30.91	29.08	67.95	67.98	28.91	27.25	26.58	24.39
Condition	hping3 –flood –rand-source -d 200 -S between VM3 and VM4							
# packets iperf	797 225		3 375 468		414 080		409 686	
Bandwidth between VM1 and VM2 (Gb/s)	23.38	23.15	66.86	66.54	25.7	24.07	16.29	17.01
Bandwidth loss %	24.36	20.39	1.60	2.12	11.10	11,67	38.71	30.26

other 96 containers targeting the 4th container resulted in an average of 3.13 Gbit/s. This considerable gap of 95,17% led to the concern that the network loss was still linked to a saturated processor.

Scenario 2 architecture (Fig. 4): To confirm the previous concern, new tests were done separating the first 4 containers one host n2, and all other containers on another host n3 that launched the attacks. As previously done an iperf was run between container 2 and 3 on the first host, under normal conditions, with an average of 68.83 Gbps similar to that of the first scenario. Then running hping on the other 96 containers to the target container 4, giving an average of 54.94 Gbps. Here a 20.18% loss resulted remaining relatively correct considering that almost 100 hpings were launched. The huge loss in scenario 1 was confirmed to be the processor being saturated with all the containers running on the same host.

B. KVM VMs Test

Scenario 1 architecture (Fig. 3) host n2: The purpose this time was to be able to run a 100 VMs, but being more resource demanding than containers, the host was quickly saturated, therefore the number of VMs was slowly increased to reach 40. Beyond that, the host was not powerful enough to run them. Doing the iperf, under normal conditions showed an average of 4,057 Gbits/s. Then hping was launched on the other VMs. Under this condition 225,7Mb/s of bandwidth resulted on average, which reflects the same behavior as the containers with a loss of 94,44%.

Scenario 2 Architecure D used the same principle as with containers the first 4 VMs were on host n2, and the rest on host n3. As observed before only 40 VMs were able to run before the processor was saturated. Running iperf under normal circumstances gave an average of 3.463 Gbit/s. Then with 40 hping an average of 3.187 Gbit/s, a loss of about 7.97%.

VI. PROBLEMS DISCOVERED

After having carried out stress tests particularly with containers, problems were found on the campus network. There are interesting flaws that could have a great impact on other physical equipment and the network itself.

A. Network Impact

First, latency issues were noticed. Whenever the stress test reached a 100 hpings, the network started to slow down significantly in the lab where the testing was carried out, and at some point the gateway was no longer responding. After doing some research with other colleagues working at the IT department, a storm-control was setup on the the switches where the testbed machines were located. This protocol made possible the detection of massive arrival of messages caused by the stress testing, blocking the flow arriving on the ports. This way when hping was launched, only a few packets were able to pass on the network with all others being rejected. In this particular case the storm-control was setup with a determined percentage each time to find the level needed to prevent attacks of this kind while avoiding reducing performance, as this protocol also affected the upload rate of the machines.

B. Hardware Impact

Another problem found was that the lab firewall was being saturated as packets were sent by hping having the random source option set up. The targeted machines tried to reply each time to each receiving message with different random addresses and even trying to reply to IPs coming from the internet. Therefore, the messages were not contained in the lab, but went through the network core up to the Internet. The firewall, seeing an answer arriving without having seen the requests going through, blocked the packets. As a result, it was denying hundreds of thousands of requests per second from the targeted machines and was logging all of them. The firewall was not configured to receive such a large number of logs and did not have the necessary memory while hundreds of gigabytes in files were generated, saturating the memory allocated. In order to avoid this problem, the IT department increased the allocated storage array for the logging as a temporary solution while a new policy was being developed to be implemented throughout the campus where necessary.

TABLE IV
HPING PERFORMANCE TEST SUMMARY

Host Config	Intel Core i7-6700 3.4GHz 8 core 32Go RAM	2x Intel Core i7-6700 3.4GHz 8 core 32Go RAM- Target & Attacker	Intel Core i7-6700 3.4GHz 8 core 32Go RAM	2x Intel Core i7-6700 3.4GHz 8 core 32Go RAM- Target & Attacker
OVS Config	1 OVS w/o VLANs	2 OVS (1 per host) w/o VLANs	1 OVS w/o VLANs	2 OVS (1 per host) w/o VLANs
Condition	Normal			
Bandwidth between VM1 and VM2 (Gb/s)	4.057	3.463	64.93	68.83
Condition	hping3 –flood -d 200 -S to VM4 from 96 VMs		hping3 –flood -d 200 -S to Container 4 from 96 Containers	
Bandwidth between VM1 and VM2 (Gb/s)	225	3.187	3.13	54.94
Bandwidth loss %	94.44	7.97	95.18	20.18

VII. SECURITY REVIEW

The main disadvantage of containers is that they can lead to security problems. By having privilege access on a container, it is possible for a container to have access to the resources of physical host and subsequently on the network as seen before with the problems encountered. Sultan et al. [24] survey the literature describing use cases that cover container security and solutions protecting them from inside attacks as well as their surrounding environment. These concerns must be addressed, and the goal is to take proper measures. Therefore we tried to implement three basic but important solutions to limit containers to have access only to the resources necessary and avoid external impact. These are RAM, CPU and rights limitation.

1) RAM Limitation: The first approach is to limit the amount of RAM allocated with a simple tool such as cgroups, a linux kernel feature that limits and isolates the resource usage of processes. This way we can simply define a threshold not to be exceeded. Using this feature we limited the containers created on a host with 32 GB of RAM down to 2GB of RAM. By performing an htop which is a linux interactive system monitor and process manager, it shows that this limit is correctly set up. In order to see if this limit is well respected and preventing the saturation of the host, a tool called stress was used allowing the saturation of either the processor or the RAM of the host on which it is used. By running stress, the allocation was not overflowing and an expected average of 2GB was obtained.

2) CPU Limitation: The second approach is the CPU limitation. As far as RAM is concerned, the restraining is quite simple. On the other hand, the processor becomes more complicated as it cannot be limited to certain percentage, e.g. going from 3.4 GHz down to 0.34 GHz. It can however be limited to the number of cores. As an example, the container was allocated the first core (n0) in the configuration file. Then, by running the stress command and processor, usage is observed. It was noted that out of 8 cores available, only one was saturated, which shows that the cgroups achieve the desired bridle. By default, the containers distribute the processor evenly. If no restriction is set, a container uses all the cores of the host. If two containers are launched without restrictions, each of them can end up using all 8 cores, resulting in a saturation of the processor, sharing about 50% of the resources.

3) Limitation of Rights: By default, on linux containers solutions, the users are created and execute containers as root. This poses security problems, as an attacker who manages to obtain rights on a container will be able to perform a privilege escalation to access the resources of the host having administration rights. One way to fix this flaw is to create a limited user specially dedicated to the environment in question with no rights other than those dedicated to its environment and not on the outside infrastructure.

VIII. CONCLUSION PERSPECTIVES AND FOR FUTURE WORKS

This work has allowed us to confirm that, when experimenting with new virtualization technologies, it is important to ensure not only the security and robustness of the tools, or the platform in which they are tested, but also the external environment in which these solutions share both physical and network resources. In spite of the robustness that virtual machines or container solutions offer, we can conclude that they are still exposed to vulnerabilities both in themselves and in the environment in which they are implementation and management.

In a first effort we have sought to understand the operation of virtualization and containers as well as the different tools by describing and comparing them. In this way we built a test environment in which cyber security attacks were performed using KVM, LXC and Docker. This testbed was installed in the Remotelabz platform in which students perform different experiments simultaneously. The main objective was to study the behavior of the tools and to observe the impact on the global network in the University during the experimentation. This testbed was divided in four architectures where 2 sets of tests were performed in which the bandwidth was measured in normal time and stress tests with hping in each of them. In the first set of tests a comparison of bandwidth between four vms and containers was made. First with KVMs and four VMs connected to a virtual OvS switch, as well as the use of host VLANs with an Intel Core i5-3230 2.6GHz 4-core

processor with 4Go RAM. During the stress tests, a bandwidth loss of 62% and 57% was observed when separating the VMS with two switches without VLANs. In a second test a host with Intel Core i7-6700 3.4GHz 8 core processor and 32Go RAM was used doing the same tests obtaining 24.32% and 25.71% of bandwidth loss respectively. In a third test a host with Intel Xeon E5-2630L1.8GHz 8 core processor with 32Go RAM host was used with VMware to be able to change the number of cores used. The same tests were performed on the same architectures using first 8 cores and then 2 cores obtaining a loss of 5.69% and 0.057% and then 47.91% and 57.27 respectively. This allowed us to conclude that the host CPU was a fundamental part in increasing the bandwidth and avoiding an important loss of bandwidth in the resistance to denial of service attacks. The same tests were performed with the other two tools obtaining a similar behavior with the difference that the containers offered a higher performance than the VMs. We also found that the use of vlans and the amount of RAM memory did not significantly impact the performance of each tool.

In the second set of tests we used the same principle as in the first set, but more attached to reality, scaling each architecture to 100 virtual machines and containers distributed firstly in a host with Intel Core i7-6700 3.4GHz8 core 32Go RAM followed by 2 hosts with the same characteristics. The loss and gain of bandwidth were increased in a considerable way obtaining differences of up to 94.44% of loss against 7.97% respectively. At this point we could observe an increase in latency and a reduction in the response of the laboratory network where the tests were performed to the point where the gateway stopped responding. Then we could see that the global network of the university was affected by saturating the main firewall with the generation of hundreds of gigabytes in logs. This prevented us from continuing our second set of tests with docker.

The security flaws found confirmed that the campus network and hardware was indeed impacted. This will allow the implementation of conception of new policies and protocols on the university's network and secure the infrastructure by collaborating with the IT department to patch up these vulnerabilities.

In a future work we will make a detailed comparison between docker unikernel-type solutions, which promise to be safer and less resource consuming than existing solutions.

Acknowledgment

The authors express their gratitude to the Mexican National Council for Science and Technology (CONACYT) for financially supporting this work through the "Conacyt-Gobierno Frances" 2017 program.

This work is also supported by a public grant overseen by the French National Research Agency (ANR) as part of the "Investissements d'avenir" program (reference: ANR-16-DUNE-0001-EOLE).

References

[1] K. Suo, Y. Zhao, W. Chen and J. Rao, "An Analysis and Empirical Study of Container Networks," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 189-197.

[2] A. A. Semnanian, J. Pham, B. Englert and X. Wu, "Virtualization Technology and its Impact on Computer Hardware Architecture," 2011 Eighth International Conference on Information Technology: New Generations, Las Vegas, NV, 2011, pp. 719-724.

[3] "The state-of-the-art in container technologies: Application, orchestration and security" E. Casalicchio S. Iannucci. Concurrency and Computation: Practice and Experience pp 5668, 2020-01-19

[4] R.J. Creasy, "The Origin of the VM/370 Time-Sharing System", IBM Journal of Research and Development, IBM, 1981, vol. 25, no. 5, pp. 483.

[5] J. E. Smith and Ravi Nair, "The architecture of virtual machines," in Computer, vol. 38, no. 5, pp. 32-38, May 2005.

[6] Roy, A., Yocum, K., and Snoeren, "Challenges in the emulation of large scale software defined networks". A. C. APSYS 2013.

[7] Pahl, Claus Brogi, Antonio Soldani, Jacopo Jamshidi, Pooyan. (2017). Cloud Container Technologies: a State-of-the-Art Review. IEEE Transactions on Cloud Computing. PP. 1-1. 10.1109/TCC.2017.2702586.

[8] Lubomski P., Kalinowski A., Krawczyk H. (2016) Multi-level Virtualization and Its Impact on System Performance in Cloud Computing. In: Gaj P., Kwiecień A., Stera P. (eds) Computer Networks. CN 2016. Communications in Computer and Information Science, vol 608. Springer, Cham

[9] A. M. Joy, "Performance comparison between Linux containers and virtual machines," 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, 2015, pp. 342-346.

[10] A Babu , Hareesh M J, J. Martin, S Cherian, Y Sastri. "System Performance evaluation of Para virtualization, Container virtualization and Full virtualization using Xen, OpenVZ and XenServer". 2014 Fourth International Conference on Advances in Computing and Communications.

[11] Y. Huang, B. Chen, M. Shih and C. Lai, "Security Impacts of Virtualization on a Network Testbed," 2012 IEEE Sixth International Conference on Software Security and Reliability, Gaithersburg, MD, 2012, pp. 71-77.

[12] S. Siraj, A. K. Gupta, I. Badgujar "Network Simulation Tools Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2012.

[13] "The comparison of network simulations for SDN". Y. Kondratyuk,University Poltava National Technical

[14] "Type 1 and Type 2 hypervisors". [Online]. Available: https://searchservervirtualization.techtarget.com/feature/Whats-thedifference-between-Type-1-and-Type-2-hypervisors.

[15] T. Kurek, "Unikernel Network Functions: A Journey Beyond the Containers," in IEEE Communications Magazine, vol. 57, no. 12, pp. 15-19, December 2019.

[16] "Remotelabz", project DUNE Eole (ANR-16-DUNE-0001-EOLE, PIA 3), CReSTIC laboratory (EA 3804), University of Reims Champagne-Ardenne.

[17] VMware [Online]. Available: https://www.vmware.com/

[18] hping - "Active Network Security Tool" [Online]. Available: www.hping.org

[19] iPerf - "The ultimate speed test tool for TCP, UDP and SCTP" [Online]. Available: www.iperf.fr

[20] Open vSwitch - "An open virtual switch" [Online]. Available: http://openvswitch.org/

[21] A. Kivity, Y. Kamay, D.Laor, U. Lublin, and A. Liguori. "KVM: the Linux virtual machine monitor". In OLS '07: The 2007 Ottawa Linux Symposium, Jul. 2007, pp. 225-230

[22] M. Uehara, "Performance Evaluations of LXC Based Educational Cloud in Amazon EC2," 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, 2016, pp. 638-643.

[23] A. Lingayat, R. R. Badre and A. Kumar Gupta, "Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine," 2018 3rd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2018, pp. 1019-1023.

[24] S. Sultan, I. Ahmad and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," in IEEE Access, vol. 7, pp. 52976-52996, 2019.