information

MDPI

Article

PocketCTF: A Fully Featured Approach for Hosting Portable Attack and Defense Cybersecurity Exercises

Stylianos Karagiannis [1,*], Christoforos Ntantogian [1], Emmanouil Magkos [1,*], Luís L. Ribeiro [2] and Luís Campos [2]

[1] Department of Informatics, Ionian University, Plateia Tsirigoti 7, 49100 Corfu, Greece; dadoyan@ionio.gr
[2] PDM&FC, R. Fradesso da Silveira, 4-1B, 1300-609 Lisboa, Portugal; luis.ribeiro@pdmfc.com (L.L.R.); luis.campos@pdmfc.com (L.C.)
* Correspondence: skaragiannis@ionio.gr (S.K.); emagos@ionio.gr (E.M.)

Abstract: Capture the flag (CTF) challenges are broadly used for engaging trainees in the technical aspects of cybersecurity, maintaining hands-on lab exercises, and integrating gamification elements. However, deploying the appropriate digital environment for conducting cybersecurity exercises can be challenging and typically requires a lot of effort and system resources by educators. In this paper, we present PocketCTF, an extensible and fully independent CTF platform, open to educators to run realistic virtual labs to host cybersecurity exercises in their classrooms. PocketCTF is based on containerization technologies to minimize the deployment effort and to utilize less system resources. A proof-of-concept implementation demonstrates the feasibility of deploying CTF challenges that allows the trainees to engage not only in offensive security but also in defensive tasks that have to be conducted during cybersecurity incidents. When using PocketCTF, educators can deploy hands-on labs, spending less time on the deployment and without necessarily having the advanced technical background to deploy complex labs and scenarios.

Keywords: CTF challenges; cyber ranges; virtual labs; security labs; blue team; virtualization

Citation: Karagiannis, S.; Ntantogian, C.; Magkos, E.; Ribeiro, L.L.; Campos, L. PocketCTF: A Fully Featured Approach for Hosting Portable Attack and Defense Cybersecurity Exercises. *Information* **2021**, *12*, 318. https://doi.org/10.3390/info12080318

Academic Editor: Sokratis Katsikas

Received: 30 June 2021
Accepted: 5 August 2021
Published: 8 August 2021

1. Introduction

Capture the flag (CTF) challenges and virtual labs for organizing hands-on cybersecurity exercises have recently become very popular worldwide [1,2]. Presented mostly during hacking conferences such as DEF CON [1] or Black Hat, CTF challenges are frequently being used by professors and other educators as well [3–8]. A significant benefit of CTF and virtual labs is the capability to replicate realistic cybersecurity scenarios that can engage students and practitioners in offensive tactics [6]. Moreover, CTF challenges can be used inside and outside of the classroom by assigning exercises to the students since self-learning attributes can be integrated [9]. However, in hosting a CTF, there seems to be a tradeoff between realistic, real-world exercises and infrastructure resources [2]. In particular, for the purpose of deploying realistic virtual labs and CTF challenges, a complex infrastructure has to be set up that must include all of the important services, from vulnerable services to defensive tools and CTF platforms to host the flags.

In the past, creating authentic computer security scenarios has been identified as a very demanding and challenging task that requires a great deal of effort from educators and lab personnel [10]. In addition, the learning outcomes from using hands-on practices need to follow curriculum guidelines or frameworks that address the required collaborative activities in cybersecurity within the industry, government, and academic institutions [2,10,11]. What is more, educators are usually in need of a way to provide customized challenges according to the courses and to privately maintain the overall learning process. In this direction, open virtual labs can better support the educator requisites, such as Vulnhub [12], which is an open repository that provides hands-on lab cybersecurity exercises as virtual images [3,11,13]. Similarly, the SEED labs [14,15] maintain virtual labs, and ENISA CSIRT [16] publishes training labs in the form of virtual images to support hands-on training sessions along with instructive materials [6]. Another important approach is Cyberdefenders [17], which is an online platform that hosts CTF challenges that mostly focus on defensive tasks and blue teaming. Finally, a deployment approach with the name DetectionLabELK [18] has been proposed by the previous developer, which is based on DetectionLab [19] for automating the creation of the virtual labs.

The aforementioned approaches can be used by educators to maintain a large variety of challenges in their classrooms. However, the methodology for how to integrate such approaches into the classroom is not always clear, and a significant set of skills is required from the educators. Consequently, the deployment complexity for educators is high, and the overall process is time-consuming and requires significant system resources [3,20–22]. Moreover, tools and services for actively detecting attacks and responding to them are usually challenging to install and configure [4]. As a result, the tools required to conduct defensive actions are usually not included in the deployment. Based on the above observations, this paper presents PocketCTF, a flexible, easy to deploy and portable platform for cybersecurity educators to create and manage virtual labs, both for training offensive and defensive security. The underpinning idea of PocketCTF is the use of LXC containers running Dockers (instead of virtual machines), in order to consume fewer system resources. By using the proposed approach, educators can easily run portable and lightweight virtual labs that can host both attack and defense scenarios. PocketCTF can manage a large variety of security tools, reducing the total effort for the educators to deploy the exercises by providing a ready-to-deploy implementation [3,23,24]. Important software packages that relate to defensive tasks are included while the vulnerable services are deployed by using containerization technologies [25]. More specifically, first a performance evaluation of various virtualization technologies is conducted including Dockers, Linux Containers (LXC) [26,27], and the Kernel-based Virtual Machine (KVM) [28–30]. The main goal is to identify the most suitable virtualization technology that would allow instructors to deploy complex scenarios without introducing significant performance overheads [1,31,32]. Next, we present the software architecture of PocketCTF and elaborate on its main components and its advantageous features. Based on this architecture, a proof-of-concept implementation was deployed in order to run a CTF scenario that relates to blue teams and defensive tactics. Overall, we argue that PocketCTF is a realistic approach that can host cybersecurity exercises and can be deployed inside a classroom or for hosting a frictionless CTF competition.

The rest of this paper is structured as follows: Section 2 presents the related work. Section 3 investigates and compares various virtualization and containerization technologies to identify their advantages and drawbacks. Section 4 analyzes the software architecture of PocketCTF and presents a proof-of-concept implementation of a CTF scenario. Section 5 evaluates the proposed approach, and Section 6 contains the conclusions and the future work.

2. Related Work

The related work in the area of CTF platforms is vast and includes many interesting works that propose various approaches to design, implement, and evaluate its role in cybersecurity education. Here, we present a subset of previous works that are the most relevant to the PocketCTF approach. The idea of using LXC containers or Dockers instead of virtual machines for CTF labs has been under research over the last few years [11,30]. More specifically, Irvine et al. introduced a framework for parameterizing cybersecurity labs by using Docker containers instead of virtualization technologies [24]. In another research study [33], a cyber range for hosting web attacks was deployed on a Raspberry Pi. The researchers demonstrated that virtual labs can be deployed by using only containerization technologies. The key benefits of using containers instead of virtual machines include the usage of fewer resources, allowing the educators to deploy a higher number of systems and services that are easier compared to virtual machines. Similarly, AlSalamah et al. [34]

analyzed how containerization techniques open new possibilities for cybersecurity educational labs, highlighting the difference between virtual machines and containers. Likewise, research has been conducted in terms of the overall architecture and the toolsets that can be used for providing educational cyberspaces and for creating hands-on lab exercises for cybersecurity [14].

Other approaches include various reinforced learning perspectives utilizing simulation and emulation processes derived from complex CTF deployments and versatile cyber ranges [35–39]. Complex CTF deployments could include workstations, firewalls, switches, web services, and tools that are frequently used by red or blue teams to create high-fidelity training environments [40–42]. Such approaches typically appear in cyber range platforms and not in CTF deployments [43,44]. What is more, network emulation has been studied along with virtualization and attack emulation for replicating realistic scenarios and for security evaluation purposes [45]. Moreover, the benefits of virtualization and Dockers have also been considered by analyzing their capabilities for network virtualization [46,47]. The networking functionalities of containerization technology in the content of lightweight virtualization instances have been under research [30,48]. Finally, the capabilities of cyber ranges to use virtualization technologies to easily recover or to reproduce laboratory scenarios have been also studied [49].

Regarding performance evaluation, research endeavors have tried to characterize the CPU and disk I/O overhead introduced by Docker [50]. Furthermore, extensive performance evaluation of the Docker containers has been also conducted, and the benefits of containerization in comparison to virtual machines have been highlighted [51,52]. Another work conducted a performance comparison between LXC and Docker containers [53]. Moreover, in our previous work [54], we performed an initial performance comparison between several virtualization and containerization technologies. Based on the results of [54], here, we select the most appropriate technology to design and implement the PocketCTF platform. In particular, in this paper, we propose the deployment of CTF scenarios using LXC containers that can run isolated instances of Linux environments, providing an individual cyberspace for each participant/trainee. What is more, our focal point relies on blue teaming and defensive actions (e.g., attack detection) in contrast to all of the previous works, which only deploy vulnerable services for offensive actions without mentioning defensive tasks.

3. Virtualization and Containerization Technologies

The main goal of PocketCTF is to deploy cybersecurity scenarios and multiple services that are required for offensive and defensive tasks without significantly increasing the total performance overhead. In this section, we discover the benefits of containerization technology and how it can serve our purposes.

3.1. Overview

Virtualization and containerization are the two most frequently used mechanisms to host applications in a computer system. Virtualization uses the notion of a virtual machine to simulate hardware devices for running a separate operating system (i.e., guest OS). There are various virtualization technologies, including the Kernel-based Virtual Machine (KVM), to run Virtual Machines.

Contrary to virtual machines, containers can be used to host the OS to run applications and services. Containers use the resource isolation features of the Linux kernel (such as cgroups and kernel namespaces), which allows them to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines. It is also possible to use containers as a lightweight virtualization option and even initiate graphical user interfaces by installing the related software packages. A significant benefit of containers compared to virtual machines is that the former is easier to manage than the latter and requires fewer system resources as well. Therefore, using containers, we can create complex network topologies requiring less deployment and integration effort. On the other hand, the main downside of containers is related to the fact that they share access to the kernel of the host, introducing security issues. Consequently, malware can gain access to the main host if it manages to escape from the container. The two basic containerization technologies are Dockers and LXC. Dockers (i.e., the most popular container technology) are lightweight, standalone, executable packages that include everything required to run applications. Docker is a single-purpose virtualization application, while LXC is multi-purpose operating system virtualization. On the contrary, LXC focuses on OS containerization being capable of deploying systems that are closer to an operating system and maintaining separate Linux kernels. In both technologies, a base host image is used to deploy the containers by including the components and software packages in the base host image. Therefore, each container is considered to be a running instance of the base host image. The main differences between Dockers and LXC are elaborated on in [55]. A summary of the differences between virtual machines and containers is presented in Table 1.

Table 1. Summary matrix for benefits and drawbacks for virtual machines and containers.

Parameter	Virtual Machines	Containers
Guest OS	Each virtual system runs on virtual hardware, and the kernel is loaded into its virtual memory.	All the guests share the same kernel loaded in the physical memory.
Isolation	Libraries and files are completely isolated.	Directories can be mounted and can be shared between the containers and the physical machine.
Performance	All instructions need to be translated between virtual systems and the physical machine, which incurs a performance decrease.	Near-native host performance and is especially better in terms of I/O interruptions.
Communication	Full virtualization of network devices.	A special driver is assigned for connecting the containers to network interfaces.
Storage	Need a large amount of disk space as each virtual system needs to store the whole OS and associated applications.	Containers require less amount of storage as the base host image is shared among the containers.
Memory requirements	Virtual machines create a unique kernel that requires a significant amount of memory	Containers do not have significant memory requirements

There is also the ability to run a virtual machine inside another virtual machine. This is called nested virtualization and offers great flexibility and cost savings. Nested virtualization is important for cloud computing, as deploying VMs on top of Infrastructure-as-a-Service cloud providers is becoming more commonplace and requires nested virtualization support [56,57]. However, poor nested virtualization performance remains a key issue. In this paper, we define the notion of nested containerization, which is the execution of a container inside another container. Combinations can be also considered, such as running a container inside a virtual machine. As we mention below, nested containerization is a key technological component in PocketCTF architecture. In Table 2, the capability of supporting nested KVM, Dockers, and LXC is presented. We observe that a KVM virtual machine can initiate all technologies. Thus, it is possible to initiate a KVM-based virtual machine inside another KVM virtual machine (nested virtualization). We also observe that it is also possible to initiate a Docker container inside another Docker instance. More importantly, we can deploy Docker containers inside an LXC. On the contrary, it is not possible to deploy an LXC container inside a Docker.

Table 2. Nested virtualization and containerization options.

Host Machine	Nested KVM	Nested Docker	Nested LXC
KVM	✔	✔	✔
Docker	—	✔	—
LXC	—	✔	—

Even if the deployment of Docker containers inside a Docker is possible, it requires specific privileges from the main Docker daemon (namely privileged mode). On the contrary, LXC can bypass the requirement for the privileged mode and uses nesting features to enable the deployment of Docker containers internally without acquiring privileges from the main host. To this end, the design principles of LXC are more focused on the creation of environments that are as close as possible to a standard Linux installation. Therefore, as a technology, LXC is much closer to a complete OS environment with attached networking and storage interfaces, while the filesystem is just an abstraction for Docker.

As mentioned above, LXC containers are not executed in a privileged mode, a feature that is required to deploy a Docker inside a Docker. This is an important concept from a security point of view, as a CTF platform is a multitenant environment, and security should be taken into consideration to avoid any malicious actions that may threaten the integrity of the whole platform. On the other hand, running LXC containers in non-privileged mode thwarts privilege escalation attacks that are possible in nested Dockers running in Dockers since the latter runs in privileged mode. Furthermore, LXC uses AppArmor [58] to ship with a default security profile intended to protect the host from misuses inside the container.

3.2. Performance Evaluation of Virtualization and Containerization Technologies

It is important to conduct a performance evaluation and to deploy test cases for investigating the benefits and drawbacks of each virtualization technology. In particular, the purpose of the experiment was to discover the performance capabilities and to measure the total overhead of each virtualization and containerization technology (with or without nested virtualization or containerization). To this end, we have considered the following scenarios: (i) a host system running Linux without virtualization; (ii) initiating a Docker container with XRDP and GUI; (iii) initiating an LXC container that hosts an operating system (also supporting XRDP and GUI); (iv) initiating a KVM based virtual machine; (v) initiating a Docker container inside an LXC container; (vi) initiating Docker services that are executed on a KVM based virtual machine; and (vii) initiating a Docker container inside another Docker.

To perform the evaluation tests, a native Linux system was used, and the specifications included an Intel Core i7-9750H CPU, 12GB DDR4 RAM and 1TB NVME-SSD. In all of our tests, we ensured that all of the other applications were turned off. In the conducted experiments, Passmark [59] was used to retrieve the CPU and memory scores, and Flexible I/O Tester (FIO) [60] was used for the disk performance tests. We have considered the following metrics in detail:

1. CPU Score: This is an aggregate mark that provides the overall CPU performance. The performance metrics are based on multiple tests including mathematical calculations, sorting, and compression algorithms, among others.
2. Memory Score: Memory performance is calculated similarly to the CPU by initiating SQL database operations, memory read of the cached and uncached RAM, calculating the memory latency, among others.
3. Disk Read and Write (MB/sec), Read and Write I/O (MB/sec): Disk performance is provided by the calculations of reading and writing speeds. Another metric is provided as well, called IOPS (Input/Output Operations Per Second), which is retrieved by the following formula: IOPS = (MBps / Block Size) * 1024. The performance test calculates the time required to iteratively read and write a big file of 800 MB for 60 s.

In Table 3, the results from the performance evaluation and benchmarks are presented. Considering the first results, the benefits of containerization included the low-performance overhead in terms of the I/O–disk cache writing and reading speeds (see Table 3). As a result, the containers can execute applications and services faster than virtualization technologies such as KVM. Moreover, executing a Docker using KVM reduces the total performance (CPU Score of 7933, 3000 lower than the other approaches). Therefore, using KVM to run containers is considered resource-intensive and eventually significantly reduces performance. Running Docker containers in LXC maintains a very good performance and is very close to the performance of the host machine. Finally, on each of the containerization deployments, the disk performance outweighs virtual machines due to the fact that the containers isolate and use a smaller amount of disk space, which is stored as a partition to the base host.

Table 3. Benchmark summary of the most popular virtualization/containerization technologies.

Performance Test	CPU Score	Memory Score	Disk Read (MB/Sec)	Disk Write (MB/Sec)	Read IOPS (MB/Sec)	Write IOPS (MB/Sec)
Native Linux	11,163	2636	114	28.5	6507	1629
Docker	11,262	2523	214	53.5	12,500	3208
LXC	11,068	2550	208	51.9	12,200	3114
KVM	7664	2530	111	27.8	6659	1667
Docker in LXC	11,318	2667	213	53.5	12,532	3136
Docker in KVM	7933	2636	100	25.2	5906	1478
Docker in Docker	11,220	2577	205	51.2	11,000	3070

Based on the above analysis, we considered the option of using LXC and Dockers running inside LXC as the most appropriate technologies for deploying the trainees' virtual hosts in PocketCTF. The deployment of LXC containers requires less deployment effort and nearly zero configuration. Moreover, LXC containers can run Dockers to install and run applications and services. In this way, the disk space requirements are reduced, resulting in better availability, extensibility, and scalability overall [48].

4. PocketCTF: A Portable Capture the Flag

The outcomes of our evaluation were important for discovering the benefits and challenges of each virtualization technology and to decide which technology would fit our needs. In this section, we present our approach based on the findings of the previous section.

4.1. Software Architecture

The architectural software components of PocketCTF are presented in Figure 1. The management service is the interface used by the educator to manage the cybersecurity exercise scenarios. Educators deploy and initiate LXC instances separately for each trainee. Note that the educator must manually inform (e.g., send an email) each trainee of their assigned IP address of their LXC instance to perform an RDP connection and to gain access to their cyberspace environment to play the CTF scenario.

According to the proposed architecture (Figure 1), the LXC instances (i.e., the Virtual Network 01, 02, etc.) hosts an Ubuntu Linux as the main environment for each trainee, featuring a GUI (gnome-desktop) including important security tools such as Suricata, Wireshark, and Nmap, among others. Each LXC container maintains the virtual Ubuntu system in a different namespace, and the vulnerable services along with the attack hosts are deployed internally to the LXC by using Docker containers. In other words, each LXC container includes several Docker containers that have a unique IP address to the virtual network. The running Dockers inside the LXC either perform attacks that the trainee should detect or include the vulnerable hosts/services. Note that these LXC containers can be converted to an LXC image to easily replicate the infrastructure. As presented in

Figure 1, for the management of the various LXC instances, an LXD server is required. The LXD server enables the easy and quick deployment of different LXC containers, each one with its own virtual network. The LXD server is deployed using Proxmox [61], which is an open-source server virtualization platform for KVM virtual machines and LXC containers with a web-based interface.

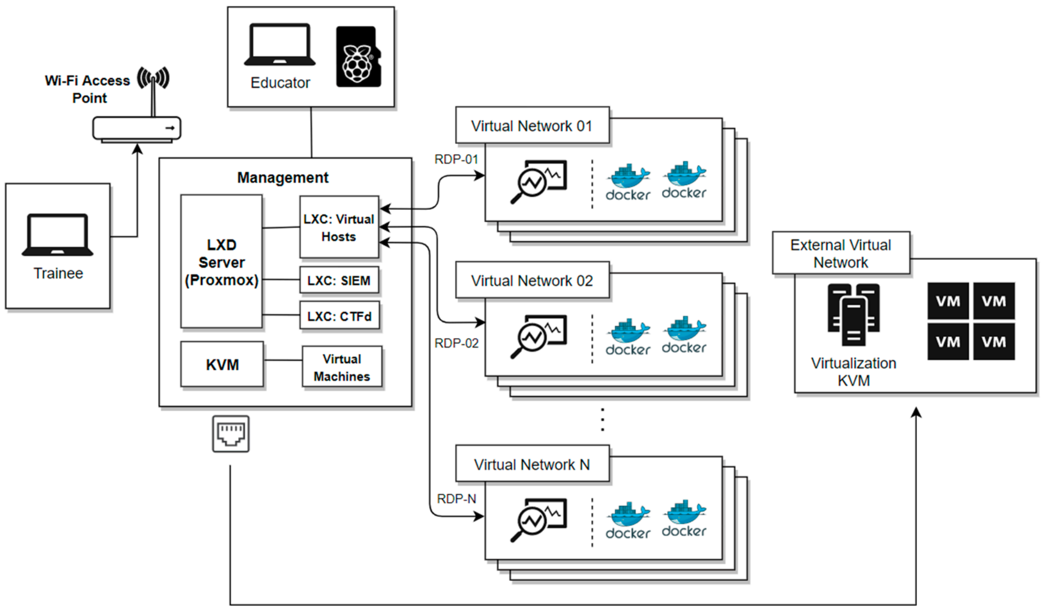


Figure 1. Software architecture of PocketCTF.

Moreover, the educator can access the frontend interface of PocketCTF to manage the exercises. This interface is based on CTFd [62] (an open-source platform that can hosts CTF challenges), which is installed on a dedicated LXC container. The PocketCTF reserves a separate LXC container that is only for running a SIEM, which is of a significantly heavyweight to be installed in separate LXC containers for each trainee. The downside of this approach is that the same instance of the running SIEM is accessed by every trainee. In PocketCTF, we have opted for the deployment of an open-source SIEM named Wazuh [63]. The proposed approach can be extended by using internal or external virtual machines that could host additional services that might require more resources or that can run Microsoft Windows virtual machines.

4.2. Implementation of a Proof-of-Concept Scenario

To implement a proof-of-concept scenario, we have deployed a CTF scenario using PocketCTF. The main learning goal of the scenario was to engage students to learn how to configure a SIEM and to detect attacks such as DoS and port scanning. This scenario requires one LXC container running the Ubuntu OS and three Docker containers. In particular, the first Docker container (Docker 01) represents the victim host, while the two other Docker containers (Docker 02–03) initiate a DoS attack and port scanning (using hping [64] and Nmap [65]). The attacks are automatically repeated every 2 min using the created shell scripts, and the detection relies on the network behavior and on the events that are extracted by Suricata. The events are parsed and sent to the SIEM, where the trainees can access and analyze the events overall. In Figure 2, the proof-of-concept deployment of the scenario is presented.

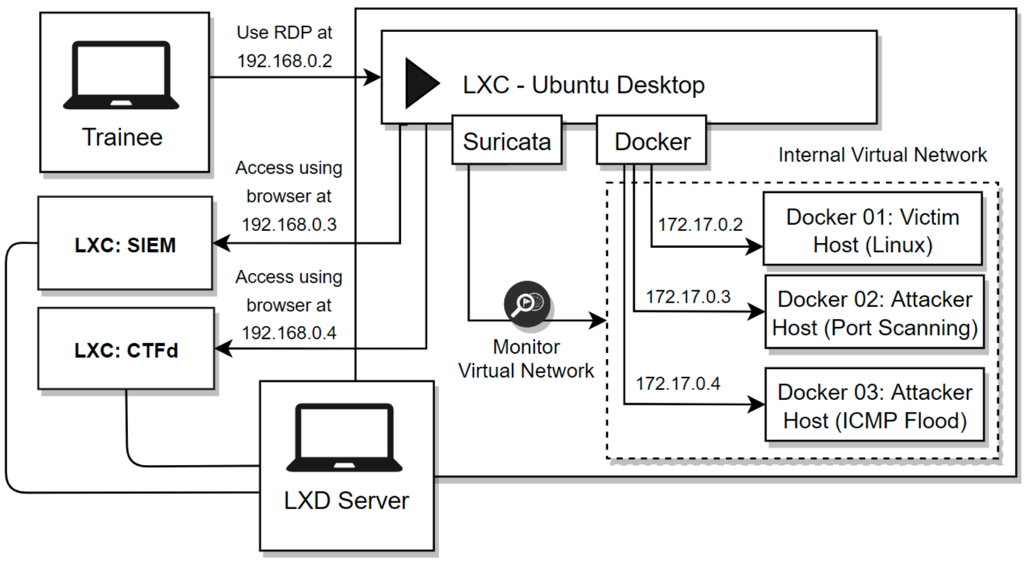
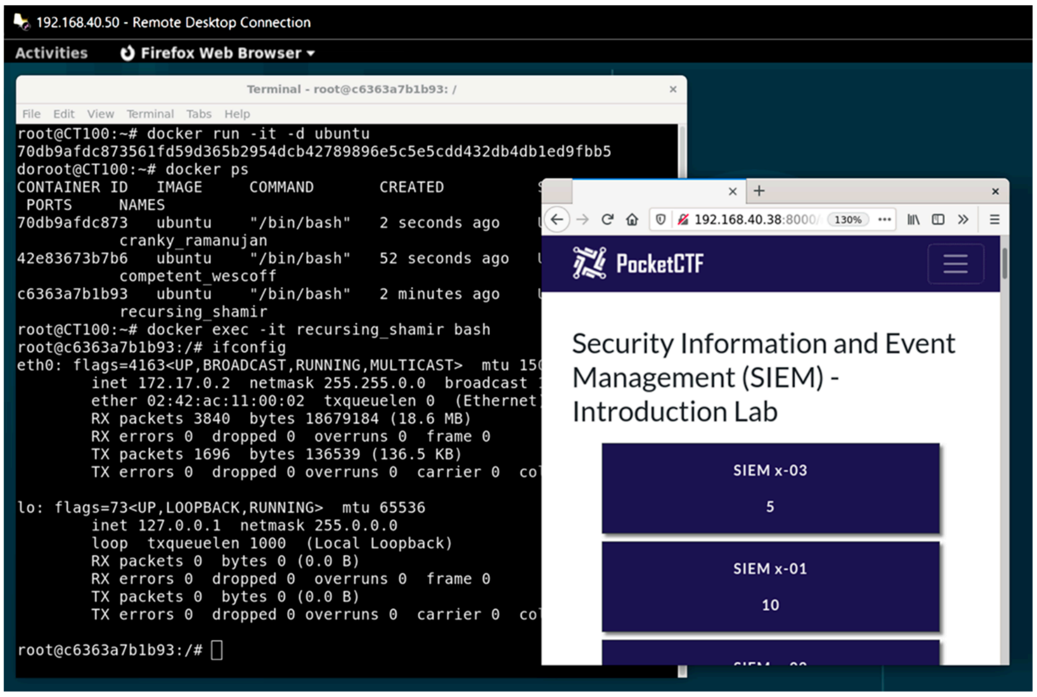

Trainee
Use RDP at 192.168.0.2
LXC - Ubuntu Desktop
Suricata
Docker
Internal Virtual Network
Access using browser at 192.168.0.3
LXC: SIEM
172.17.0.2
Docker 01: Victim Host (Linux)
172.17.0.3
Docker 02: Attacker Host (Port Scanning)
Access using browser at 192.168.0.4
LXC: CTFd
Monitor Virtual Network
172.17.0.4
Docker 03: Attacker Host (ICMP Flood)
LXD Server

Figure 2. Proof of concept deployment of PocketCTF.

Using RDP, the trainees are connected to the IP address of the LXC container that hosts the different Docker services and the required software packages. As mentioned before, the internal virtual network is handled by the LXC container, and therefore, each deployment is isolated. An example of a successful RDP connection to an LXC container that initiates a virtualized operating system is presented in Figure 3.


192.168.40.50 - Remote Desktop Connection
Activities
Firefox Web Browser
Terminal - root@c6363a7b1b93: /
File Edit View Terminal Tabs Help
192.168.40.38:8000/
130%
PocketCTF

```
root@CT100:~# docker run -it -d ubuntu
70db9afdc873561fd59d365b2954dcb42789896e5c5e5cdd432db4db1ed9fbb5
doroot@CT100:~# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED
 PORTS         NAMES
70db9afdc873   ubuntu    "/bin/bash"   2 seconds ago
               cranky_ramanujan
42e83673b7b6   ubuntu    "/bin/bash"   52 seconds ago
               competent_wescoff
c6363a7b1b93   ubuntu    "/bin/bash"   2 minutes ago
               recursing_shamir
root@CT100:~# docker exec -it recursing_shamir bash
root@c6363a7b1b93:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 15
        inet 172.17.0.2  netmask 255.255.0.0  broadcast
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet
        RX packets 3840  bytes 18679184 (18.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1696  bytes 136539 (136.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  co

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  co

root@c6363a7b1b93:/# 
```

Security Information and Event Management (SIEM) - Introduction Lab

SIEM x-03
5

SIEM x-01
10

Figure 3. Accessing the LXC containers using RDP.

The trainees must execute the following steps to complete the scenario:

1. Trainees need to access the LXC dedicated to them by using RDP: The trainees are invited to access the system via RDP. The educator may explain that this will be their main system for the exercises and the total approach.
2. Access the CTF platform to access the instructions: The trainees are invited to access the web API of the platform where the main features of the platform are explained.

3. Set up the SIEM agents: The trainees must follow a specific process to install and configure the SIEM agents. The SIEM agents are parsers with the main purpose of retrieving log files to the SIEM (as mentioned above, in PocketCTF, we have deployed Wazuh SIEM).
4. Configure IDS and read alerts: The trainees should read and evaluate the alerts from the IDS by checking the log file generated by Suricata (i.e., fast.log). Afterwards, this file will be forwarded to the SIEM.
5. Forwarding log files to SIEM for parsing: The trainees should configure the SIEM agents to ingest the fast.log (generated by Suricata) and the generic log files (generated by the host OS) into Wazuh.
6. Create scheduled queries which will trigger the corresponding alerts: The trainees must create scheduled queries that will fetch data to detect the DoS attack and generate an alert.

By following and completing the proof-of-concept scenario, the trainees are able to understand the fundamental concepts of data shippers by deploying the SIEM agents and to technically investigate the logs from the intrusion detection. The trainees are able to follow step-by-step instructions from the CTF platform in order to detect and respond to the attack in real-time.

5. Discussion

The main benefit of the proposed PocketCTF is the high performance and fewer requirements, due to the kernelless deployment of the LXC containers. Another important benefit is the small disk space required to run the containers in contrast to virtual machines. The above is possible since the LXC images (also called LXC templates) can be used to deploy multiple LXC containers that minimize the required disk space. Educators are able to create customized exercises and to store the LXC images or distribute them online. Therefore, every LXC image can maintain a different cybersecurity scenario, including multiple attack paths, and have the software components for the defensive actions included.

The benefit for the trainees is that they maintain a single point of interaction by entering the appointed LXC that maintains everything that is required. Once converted to an LXC image, the LXC container will be used as the core for deploying the LXC containers. Educators can use existing exercises that are hosted as virtual machine images (e.g., from Vulnhub), as it is possible to convert them to LXC images. It is also important to note also that we tested the possibility of deploying PocketCTF in a Raspberry Pi 4 board. The benefits and challenges for deploying in a Raspberry Pi compared to a typical PC machine are summarized in Table 4.

Table 4. Benefits and limitations of using a typical x86 compatible PC machine and Raspberry Pi.

Architecture	Benefits	Limitations
x86-compatible PC machine	Can support many trainees and has a better performance overall.	Increased cost and less portability
Raspberry Pi (ARM)	Easy to afford. Portability	Docker images might require conversion to ARM architecture by using Docker buildx [66]. KVM is too resource-intensive for RPi to run smoothly

In general, it was feasible to host the LXC containers on a Raspberry Pi 4 but with some restrictions, as discussed in Table 4. Deploying the approach on a Raspberry Pi, for example, restricted the deployment of the Dockers that are only designed for ARM technology. However, it is possible to rebuild existing Docker images for ARM technology using Docker buildx [66]. Using this tool, we successfully deployed more than 10 containers by using an Ubuntu system as the LXD host operating system on the Raspberry Pi.

In Table 5, a summary of the benefits and challenges of PocketCTF are presented with regard to portability, scalability, compatibility, and usability. The numerous benefits of our approach underscore the practical added value of PocketCTF.

Table 5. Benefits and challenges of PocketCTF.

Characteristic	Benefits	Challenges
Portability	Running Dockers inside LXC containers allows a large variety of services within a single virtual image.	The image size could be large depending on the installed services. Need for a PC with moderate resources to run multiple machines.
Scalability	Due to clustering support mode, LXC containers offer high scalability.	The deployed SIEM is shared between all participants.
Compatibility	All the popular services that have already published Docker containers can be deployed. Virtual images could be revised and executed as containers.	Windows are not supported unless KVM is initiated. ARM-based dockers must be created when Raspberry Pi is used.
Usability	Easy deployment of PocketCTF. For every new trainee, it is easy to initiate new LXC containers. Trainees and educators maintain a single endpoint of communication that automatically deploys the trainees' operating systems, CTF platform, and other services (e.g., the SIEM)	Creating or importing new PocketCTF scenarios is relatively complex.

The compatibility issues and, more specifically, the inability to deploy Windows hosts should be taken into serious consideration. A possible solution is to utilize external networks to regularly initiate Windows hosts by using KVM or other virtualization technology. Such an approach may solve this issue, but it is the opposite direction to having a portable and performance-wise solution.

6. Conclusions and Future Work

This paper presented PocketCTF, a flexible and portable platform for cybersecurity educators to create and manage virtual labs, both for offensive and defensive security training. PocketCTF can host and manage a large variety of security tools that are ready to deploy, decreasing the total effort needed to deploy the exercises. PocketCTF is focused on portability and easy-deployment and introduces a way to include all of the important software packages for hands-on labs inside a single virtual machine, while vulnerable services are deployed as Docker containers along with the victim and attack hosts. We also discussed the potential benefits of using containerization instead of virtualization technologies to deploy virtual labs. We investigated the various features that containerization and virtualization provide by analyzing the benefits of each approach. The findings indicate that deploying the required services in LXC and Docker containers will significantly decrease the total overhead. The deployment of PocketCTF confirms that the total overhead is decreased by using containers instead of virtual machines and that the total management is easier for creating and deploying cybersecurity hands-on labs. The required services necessary for the intrusion detection, the SIEM, and the CTF platform are automatically deployed or replicated and require less effort and system resources than other approaches.

Future work includes the extensive validation of PocketCTF by conducting stress tests and by including other security scenarios as well. In terms of the evaluation, we intend to further test the approach by deploying an LXD cluster to distribute the required resources and scale the approach on multiple systems. Furthermore, future work includes the conversion of existing challenges to containers and providing specific steps that have to be followed as documentation. Finally, we consider the alignment of the exercises to cybersecurity education frameworks as an important aspect. Towards this direction, we intend to further investigate the existing taxonomies regarding the learning impact by utilizing scenarios that will be executed and hosted accordingly to PocketCTF.

Author Contributions: Conceptualization, S.K. and C.N.; methodology, S.K. and C.N.; software, S.K.; validation, S.K., C.N. and L.L.R.; formal analysis, S.K.; investigation, S.K. and L.L.R.; resources, L.C.; data curation, S.K., C.N.; writing—original draft preparation, S.K.; writing—review and editing, C.N. and E.M.; visualization, S.K.; supervision, E.M.; project administration, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is performed as part of the SPHINX project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 826183 on Digital Society, Trust & Cyber Security E-Health, Well-being, and Ageing. The funding body has not participated in the elaboration of this research paper. This work was also supported by the European Commission under the Horizon 2020 Programme (H2020) through the Project CyberSec4Europe (Grant Agreement no. 830929).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taylor, C.; Arias, P.; Klopchic, J.; Matarazzo, C.; Dube, E. CTF: State-of-the-Art and building the next generation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*; 2017. Available online: https://www.usenix.org/conference/ase17/workshop-program/presentation/taylor (accessed on 4 August 2021).
2. Davis, A.; Leek, T.; Zhivich, M.; Gwinnup, K.; Leonard, W. The Fun and Future of CTF. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, San Diego, CA, USA, 18 August 2014.
3. Schreuders, Z.C.; Shaw, T.; Shan-A-Khuda, M.; Ravichandran, G.; Keighley, J.; Ordean, M. Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events. *Ase'17*. 2017. Available online: https://www.usenix.org/conference/ase17/workshop-program/presentation/schreuders (accessed on 4 August 2021).
4. Iannacone, M.D.; Bridges, R.A. Quantifiable & Comparable Evaluations of Cyber Defensive Capabilities: A Survey & Novel, Unified Approach. *arXiv* **2019**, arXiv:1902.00053.
5. Vykopal, J.; Svabensky, V.; Chang, E.C. Benefits and Pitfalls of Using Capture The Flag Games in University Courses. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 752–758. [CrossRef]
6. Mirkovic, J.; Peterson, P.A.H. Class Capture-the-Flag Exercises. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, San Diego, CA, USA, 18 August 2014.
7. Vigna, G.; Borgolte, K.; Corbetta, J.; Doupe, A.; Fratantonio, Y.; Invernizzi, L.; Kirat, D.; Shoshitaishvili, Y. Ten Years of iCTF: The Good, The Bad, and The Ugly. In Proceedings of the 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education, San Diego, CA, USA, 18 August 2014.
8. Jones, K.S.; Namin, A.S.; Armstrong, M.E. The core cyber-defense knowledge, skills, and abilities that cybersecurity students should learn in school: Results from interviews with cybersecurity professionals. *ACM Trans. Comput. Educ.* **2018**, *18*, 1–12. [CrossRef]
9. Barth, F.; Luft, M. Towards a practical approach for teaching IT-security. In Proceedings of the 3rd International Conference on Society and Information Technologies, Orlando, FL, USA, 25–28 March 2012; pp. 300–305.
10. Gondree, M.; Peterson, Z.N.J.; Denning, T. Security through play. *IEEE Secur. Priv.* **2013**, *11*, 64–67. [CrossRef]
11. Perrone, G.; Romano, S.P. The docker security playground: A hands-on approach to the study of network security. In Proceedings of the 2017 Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, USA, 25–28 September 2017. [CrossRef]
12. VulnHub. Available online: https://www.vulnhub.com/ (accessed on 24 July 2021).
13. Karlov, A.A. Virtualization in education: Information Security lab in your hands. *Phys. Part. Nucl. Lett.* **2016**, *13*, 640–643. [CrossRef]
14. Du, W. SEED: Hands-on lab exercises for computer security education. *IEEE Secur. Priv.* **2011**, *9*, 70–73. [CrossRef]
15. SEED Labs. Available online: https://seedsecuritylabs.org/ (accessed on 24 July 2021).
16. ENISA CSIRT—Training Resources. Available online: https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material (accessed on 24 July 2021).
17. CyberDefenders. Available online: https://cyberdefenders.org/ (accessed on 24 July 2021).
18. DetectionLabELK. Available online: https://github.com/cyberdefenders/DetectionLabELK (accessed on 24 July 2021).
19. DetectionLab. Available online: https://github.com/clong/DetectionLab (accessed on 24 July 2021).

20. de Leon, D.C.; Goes, C.E.; Haney, M.A.; Krings, A.W. ADLES: Specifying, deploying, and sharing hands-on cyber-exercises. *Comput. Secur.* **2018**, *74*, 12–40. [CrossRef]
21. Braidley, S. Extending Our Cyber-Range CYRAN with Social Engineering Capabilities. September 2016. Available online: https://www.researchgate.net/profile/Sam-Braidley/publication/313241265_Extending_Our_Cyber-Range_CYRAN_with_Social_Engineering_Capabilities/links/5893764445851563f828eb20/Extending-Our-Cyber-Range-CYRAN-with-Social-Engineering-Capabilities.pdf (accessed on 4 August 2021).
22. Brynielsson, J.; Franke, U.; Tariq, M.A.; Varga, S. Using cyber defense exercises to obtain additional data for attacker profiling. In Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI), Tucson, AZ, USA, 28–30 September 2016; pp. 37–42. [CrossRef]
23. Childers, N.; Boe, B.; Cavallaro, L.; Cavedon, L.; Cova, M.; Egele, M.; Vigna, G. Organizing large scale hacking competitions. In *Detection of Intrusions and Malware, and Vulnerability Assessment DIMVA 2010*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6201, pp. 132–152. [CrossRef]
24. Irvine, C.E.; Michael, F.; Khosalim, J. Labtainers: A Framework for Parameterized Cybersecurity Labs Using Containers. 2017. Available online: http://hdl.handle.net/10945/56211 (accessed on 4 August 2021).
25. Docker. Available online: https://www.docker.com/ (accessed on 24 July 2021).
26. LXC—Linux Containers. Available online: https://linuxcontainers.org/ (accessed on 24 July 2021).
27. Dua, R.; Raja, A.R.; Kakadia, D. Virtualization vs containerization to support PaaS. In Proceedings of the 2014 IEEE International Conference on Cloud Engineering, Boston, MA, USA, 11–14 March 2014; pp. 610–614. [CrossRef]
28. Hickman, A. Container Intrusions: Assessing the Efficacy of Intrusion Detection and Analysis Methods for Linux Container Environments. *SANS Inst. InfoSec Read. Room* **2017**, 1–32. Available online: https://www.researchgate.net/profile/Tiago-Heinrich/publication/346246313_Deteccao_de_Anomalias_Estudo_de_Tecnicas_de_Identificacao_de_Ataques_em_um_Ambiente_de_Conteiner/links/5fbd0f8f458515b79765b64e/Deteccao-de-Anomalias-Estudo-de-Tecnicas-de-Identificacao-de-Ataques-em-um-Ambiente-de-Conteiner.pdf (accessed on 4 August 2021).
29. Špaček, F.; Sohlich, R.; Dulík, T. Docker as platform for assignments evaluation. *Procedia Eng.* **2015**, *100*, 1665–1671. [CrossRef]
30. Yin, Y.; Shao, Y.; Wang, X.; Su, Q. A Flexible Cyber Security Experimentation Platform Architecture Based on Docker. In Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 22–26 July 2019; pp. 413–420. [CrossRef]
31. Hay, B.; Dodge, R.; Nance, K. Using virtualization to create and deploy computer security lab exercises. *IFIP Int. Fed. Inf. Process.* **2008**, *278*, 621–635. [CrossRef]
32. Raj, A.S.; Alangot, B.; Prabhu, S.; Achuthan, K. Scalable and lightweight CTF infrastructures using application containers. In Proceedings of the 2016 USENIX Workshop on Advances in Security Education (ASE 16), Austin, TX, USA, 9 August 2016.
33. Oh, S.K.; Stickney, N.; Hawthorne, D.; Matthews, S.J. Teaching Web-Attacks on a Raspberry Pi Cyber Range. In *Proceedings of the 21st Annual Conference on Information Technology Education (SIGITE '20)*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 324–329. [CrossRef]
34. AlSalamah, A.K.; Cámara, J.M.S.; Kelly, S. Applying virtualization and containerization techniques in cybersecurity education. In Proceedings of the 34th Information Systems Education Conference (ISECON 2018), San Antonio, TX, USA, 6 April 2018; pp. 1–14.
35. Vykopal, J.; Ošlejšek, R.; Čeleda, P.; Vizváry, M.; Tovarňák, D. KYPO cyber range: Design and use cases. In *Proceedings of the 12th International Conference on Software Technologies (ICSOFT 2017)*; SciTePress: Madrid, Spain, 2017; pp. 310–321. [CrossRef]
36. Pham, C.; Tang, D.; Chinen, K.I.; Beuran, R. CyRIS: A cyber range instantiation system for facilitating security training. In *Proceedings of the Seventh Symposium on Information and Communication Technology (SoICT '16)*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 251–258. [CrossRef]
37. Beuran, R.; Pham, C.; Tang, D.; Chinen, K.i.; Tan, Y.; Shinoda, Y. Cytrone: An integrated cybersecurity training framework. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy—ICISSP*; SciTePress: Porto, Portugal, 2017; pp. 157–166. [CrossRef]
38. Chouliaras, N.; Kittes, G.; Kantzavelou, I.; Maglaras, L.; Pantziou, G.; Ferrag, M.A. Cyber ranges and testbeds for education, training, and research. *Appl. Sci.* **2021**, *11*, 1809. [CrossRef]
39. Vekaria, K.B.; Calyam, P.; Wang, S.; Payyavula, R.; Rockey, M.; Ahmed, N. Cyber Range for Research-Inspired Learning of 'Attack Defense by Pretense' Principle and Practice. *IEEE Trans. Learn. Technol.* **2021**, *50*, 1. [CrossRef]
40. Costa, G.; Russo, E.; Armando, A. Automating the Generation of Cyber Range Virtual Scenarios with VSDL. 2020. Available online: https://arxiv.org/abs/2001.06681 (accessed on 4 August 2021).
41. Chaskos, E.C. *Cyber-Security Training: A Comparative Analysis of Cyber- Ranges and Emerging Trends*; Technology Development for Security Practitioners; Springer: Basle, Switzerland, 2021; p. 78.
42. Vykopal, J.; Vizvary, M.; Oslejsek, R.; Celeda, P.; Tovarnak, D. Lessons learned from complex hands-on defence exercises in a cyber range. In Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, USA, 18–21 October 2017; pp. 1–8. [CrossRef]
43. Jamalpur, S.; Navya, Y.S.; Raja, P.; Tagore, G.; Rao, G.R.K. Dynamic Malware Analysis Using Cuckoo Sandbox. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 1056–1060. [CrossRef]

44. Keahey, K.; Doering, K.; Foster, I. From sandbox to playground: Dynamic virtual environments in the grid. In Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA, USA, 8 November 2004; Volume 3, pp. 34–42. [CrossRef]
45. Chen, Y.; Zhang, G.; Hu, D.; Tao, Q. Multiscale emulation technology based on the integration of virtualization, physical and simulation networks. In Proceedings of the 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), Hangzhou, China, 23–25 June 2019; pp. 396–402. [CrossRef]
46. Song, H.; Wang, X.; Zhai, M.; Zhang, G. High-fidelity router emulation technologies based on multi-scale virtualization. *Information* **2020**, *11*, 47. [CrossRef]
47. Yang, S.; Wang, X.; Wang, X.; An, L.; Zhang, G. High-performance docker integration scheme based on OpenStack. *World Wide Web* **2020**, *23*, 2593–2632. [CrossRef]
48. Ji, Y.; Zhang, G.; Xie, S.; Wang, X. Container Networking Performance Analysis for Large-Scale User Behavior Simulation. *J. Comput. Commun.* **2019**, *7*, 136–146. [CrossRef]
49. Zhang, Z.; Lu, G.; Zhang, C.; Gao, Y.; Wu, Y.; Zhong, G. CyFRS: A Fast Recoverable System for Cyber Range Based on Real Network Environment. In Proceedings of the 2020 Information Communication Technologies Conference (ICTC), Nanjing, China, 29–31 May 2020; pp. 153–157. [CrossRef]
50. Casalicchio, E.; Perciballi, V. Measuring Docker performance: What a mess!!! In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (ICPE '17 Companion)*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 11–16. [CrossRef]
51. Bhimani, J.; Yang, Z.; Mi, N.; Yang, J.; Xu, Q.; Awasthi, M.; Pandurangan, R.; Balakrishnan, V. Docker container scheduler for I/O intensive applications running on NVMe SSDs. *IEEE Trans. Multi-Scale Comput. Syst.* **2018**, *4*, 313–326. [CrossRef]
52. Felter, W.; Ferreira, A.; Rajamony, R.; Rubio, J. An updated performance comparison of virtual machines and Linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA, 29–31 March 2015; pp. 171–172. [CrossRef]
53. Putri, A.R.; Munadi, R.; Negara, R.M. Performance analysis of multi services on container Docker, LXC, and LXD. *Bull. Electr. Eng. Inform.* **2020**, *9*, 2008–2011. [CrossRef]
54. Karagiannis, S.; Magkos, E.; Ntantogian, C.; Ribeiro, L.L. Sandboxing the Cyberspace for Cybersecurity Education and Learning. In *European Symposium on Research in Computer Security*; Springer: Cham, Switzerland, 2020; pp. 181–196. [CrossRef]
55. Moravcik, M.; Segec, P.; Kontsek, M.; Uramova, J.; Papan, J. Comparison of LXC and Docker Technologies. In Proceedings of the 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), Košice, Slovenia, 12–13 November 2020; pp. 481–486. [CrossRef]
56. Beham, M.; Vlad, M.; Reiser, H.P. Intrusion detection and honeypots in nested virtualization environments. In Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, Hungary, 24–27 June 2013. [CrossRef]
57. Ben-Yehuda, M.; Day, M.D.; Dubitzky, Z.; Factor, M.; Har'El, N.; Gordon, A.; Liguori, A.; Wasserman, O.; Yassour, B.A. The turtles project: Design and implementation of nested virtualization. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10), Vancouver, BC, Canada, 4–6 October 2010; pp. 423–436.
58. AppArmor. Available online: https://gitlab.com/apparmor/apparmor (accessed on 28 July 2021).
59. Passmark—Linux Performance Test. Available online: https://www.passmark.com/products/pt_linux/index.php (accessed on 12 July 2021).
60. FIO—Flexible I/O Tester. Available online: https://github.com/axboe/fio (accessed on 12 July 2021).
61. Proxmox. Available online: https://www.proxmox.com/en/ (accessed on 24 July 2021).
62. CTFd. Available online: https://github.com/CTFd/CTFd (accessed on 24 July 2021).
63. Wazuh—The Open Source Security Platform. Available online: https://github.com/wazuh/wazuh (accessed on 24 July 2021).
64. hping—Network Tool. Available online: https://github.com/antirez/hping (accessed on 24 July 2021).
65. Nmap—The Network Mapper. Available online: https://github.com/nmap/nmap (accessed on 24 July 2021).
66. Docker Buildx. Available online: https://github.com/docker/buildx (accessed on 24 July 2021).