

# Sandboxing the Cyberspace for Cybersecurity Education and Learning

Stylianos Karagiannis<sup>1</sup>[0000-0001-9571-4417], Emmanouil Magkos<sup>1</sup>[0000-0002-5922-4274]  
Christoforos Ntantogian<sup>1</sup>[0000-0002-1575-4572] and Luís L. Ribeiro<sup>2</sup>

<sup>1</sup> Department of Informatics, Ionian University, Plateia Tsirigoti 7, 49100, Corfu, Greece  
{skaragiannis, emagos, dadoyan}@ionio.gr

<sup>2</sup> PDM&FC, R. Fradesso da Silveira, 4-1B, 1300-609 Lisboa, Portugal  
{luis.ribeiro}@pdmfc.com

**Abstract.** Deploying the appropriate digital environment for conducting cybersecurity exercises can be challenging and typically requires a lot of effort and system resources. Usually, for deploying vulnerable webservices and setting up labs for hands-on cybersecurity exercises to take place, more configuration is required along with technical expertise. Containerization techniques and solutions provide less overhead and can be used instead of virtualization techniques to revise the existing approaches. Furthermore, it is important to sandbox or replicate existing systems or services for the cybersecurity exercises to be realistic. To address such challenges, we conducted a performance evaluation of some of the existing deployment techniques to analyze their benefits and drawbacks. We tested techniques relevant to containerization or MicroVMs that include less overhead instead of the regular virtualization techniques to provide meaningful and comparable results from the deployment of scalable solutions, demonstrating their benefits and drawbacks. Finally, we presented a use case for deploying cybersecurity exercises that requires less effort and moderate system resources and an approach for monitoring the progress of the participants using a host-based intrusion system.

**Keywords:** Cybersecurity, Docker, Sandbox, Security Labs, Cyber Range

## 1 Introduction

Designing and deploying effective cybersecurity labs requires a combination of various technologies which usually include a lot of effort for the deployment of effective cybersecurity exercises. For computer security students to benefit from hands-on experiences, a large variety of security tools must be used, making difficult and time-consuming the task of properly designing and deploying the exercises [1, 2, 3]. Creating authentic computer security scenarios has been identified in the past as a very demanding and challenging task, requiring much effort from the instructor and the lab personnel. Virtualization technologies provide beneficial ways for hosting multiple machines within one single system, decreasing the required deployment effort and system resources, enhancing the instructor's ability to deploy complex scenarios for education purposes [4]. Our purpose is to create a flexible and portable solution without requiring any existing deployed infrastructure and to deploy multiple systems for

conduct security testing that includes complex processes such as adversary emulations and incident response. Existing cybersecurity exercises are usually deployed using virtualization and restrict the learning processes without maintaining significant interactions between the deployed services (e.g. interactions between Intrusion detection systems, adversaries, and the usage of Elastic Search). Not only this but virtual machines include a lot of overhead and a high demand in system resources for deploying relevant services that are required by the exercises. On the other hand, containers have several advantages [5, 6, 7, 8]. The performance that containers have, comes with the cost of providing less isolation than virtual machines. Restrictions also apply in terms of compatibility for deploying kernelless operating systems using containerization.

A main difficulty that persists is to match the security scenarios and the required infrastructure to specific knowledge areas and technical topics. In order to identify the knowledge areas that are relevant during the exercises, it is considered important to establish well-defined taxonomies that address the acquired knowledge and skills [9]. For example, Security Operations Center (SOC) teams are meant to offer high quality IT-security services using tools that actively detect potential threats and attacks and respond accordingly [10]. In such cases, not only the deployment effort is big, but for each participant it is best to have an individual cyberspace as the environment that includes the deployed systems and services. Therefore, scalability issues derive from the fact that it is required to replicate each cyberspace for each participant. Usually, the tools required for conducting such tasks are complex and their deployment is time-consuming [11]. Finally, the learning outcomes of using hands-on practices need to follow curriculum guidelines or frameworks, which address the collaborative activities that are required in cybersecurity within the industry, government and academic organizations [2, 12, 13].

Modern technologies for deploying services or operating systems, include docker<sup>1</sup> containers, Linux Containers (LXC), MicroVMs<sup>2</sup>, RancherVM<sup>3</sup> and other options for deploying and running Kernel-based Virtual Machine (KVM) or docker containers inside a docker. Current operating systems and especially Linux distributions enhance the ability for portable and flexible deployments. Therefore, the existing exercises and tools can be easier deployed and managed accordingly.

There are specific benefits and drawbacks from using the existing technologies for deploying systems and services. This research aspires to analyze the state-of-the-art approaches for deploying cyber security exercises, considering the portability, flexibility and capability of featuring easy-deployment and to reduce the total overhead reducing the requirement in system resources. Our intention is to deploy, evaluate and investigate the best practices for using such technologies to maintain cybersecurity exercises and hands-on labs, while requiring less deployment effort. Therefore, the main research questions that this work attempts to answer are reflected below:

---

<sup>1</sup> <https://docker.com/>

<sup>2</sup> <https://github.com/firecracker-microvm/firecracker/>

<sup>3</sup> <https://github.com/rancher/vm/>

**RQ1:** What are the features, challenges and drawbacks that the different virtualization or containerization technologies include for designing and deploying cybersecurity exercises?

**RQ2:** Which are the best practices for deploying complex cybersecurity exercises while maintaining the least overhead in terms of resources and having increased compatibility?

RQ1 and RQ2 intend to evaluate the current deployment options using sandboxing for maintaining cybersecurity exercises and to discover the current possibilities of containerization and virtualization technologies. Towards this direction, we conducted an in-depth analysis and a performance evaluation of the most common technologies, while we also deployed example exercises accordingly for discovering the benefits and drawbacks for each approach. The research paper is organized as follows: In section the related work is provided, while in section 3 common virtualization technologies and containerization approaches are analyzed, presenting the capability for using a sandbox as the main learning environment. In section 4 the potential a solution of using a sandbox for maintaining complex Cyber Ranges is discussed, concluding with section 5 by discussing future action points.

## 2 Related Work

The idea of using LXC or docker containers instead of virtual machines has been under research during the last years [14, 15]. For example, Irvine et al introduced a framework for parameterizing cybersecurity labs using containers [2]. The key benefits from using containers instead of virtual machines include the higher performance that containers score, which allows to deploy a high number of systems and services more easily, requiring less resources. For example, AlSalamah et al. [13], analyze how containerization techniques could open new possibilities, highlighting the difference between virtual machines and containers. They assess the benefits that containers provide regarding configuration, networking and performance, as well as their flexibility for deploying large number of services. Likewise, research has been conducted in terms of the design of architectures and toolsets for providing learning cyberspaces related to network security and for creating hands-on lab exercises [16]. In their research the significant benefits of using dockers instead of virtualization technologies are also highlighted.

Significant work has been done in the past in providing security education hands-on labs including practical cybersecurity exercises. For example, SEED labs<sup>4</sup> provided pre-built virtual machine images including about 30 exercises and featuring a wide range of cybersecurity topics [17]. Similarly, the ENISA's Computer Security and Incident Response Team (CSIRT)<sup>5</sup>, since 2008, released and introduced training material that is constantly updated by new exercise scenarios containing toolsets and virtual images to support hands-on training sessions.

---

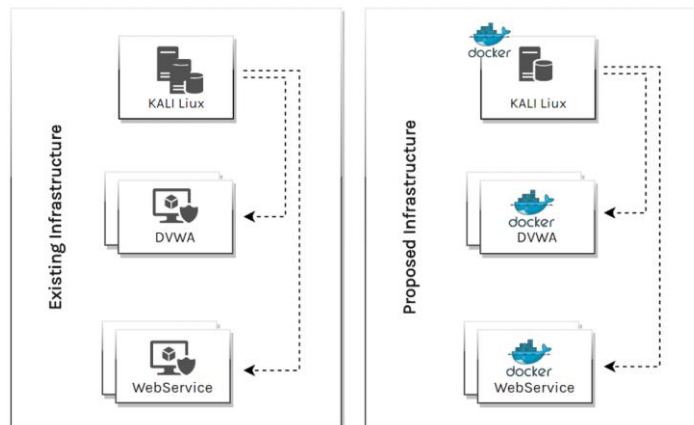
<sup>4</sup> <https://seedsecuritylabs.org/>

<sup>5</sup> <https://enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/>

Other similar approaches include reinforced learning approaches that include simulation and emulation processes derived from complex deployments. Such approaches include important elements that enterprise networks have such as workstations, firewalls and servers, among others for creating a high-fidelity training environment [18, 19, 20]. Instead, deployment options that include more complex infrastructures and network topologies are not very frequently found in CtF (Capture the Flag) exercises and are usually related to cyber ranges, where more complex topologies are presented [21, 22]. As a result, the deployment options for cybersecurity exercises are currently revised for using containerization technologies along with virtualization technologies to extend and provide more interactive cybersecurity learning environments.

### 3 Virtualization Technologies and Sandboxing

Virtualization technologies are frequently used for creating and deploying vulnerable virtual systems for testing purposes. Popular approaches include HackTheBox<sup>6</sup>, TryHackMe<sup>7</sup> and the vulnerable images published on VulnHub<sup>8</sup>, an open repository providing hands-on lab cybersecurity exercises.



**Fig. 1.** Dockerization of existing services and vulnerable systems

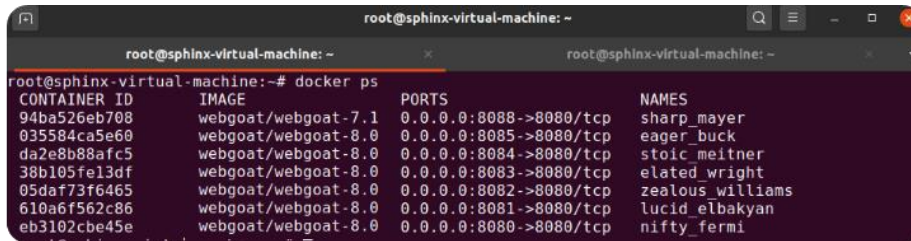
Similarly, SEED labs and ENISA CSIRT introduced training material containing cybersecurity exercise scenarios in the form of Virtual Images to support hands-on training sessions. The benefits of using virtualization techniques are many, however, the total performance and size overhead could be difficult to manage. Therefore, specific cybersecurity scenarios can be revised to reduce the total overhead accordingly. For example, existing services are possible to be revised and deployed as docker

<sup>6</sup> <https://hackthebox.eu/>

<sup>7</sup> <https://tryhackme.com/>

<sup>8</sup> <https://vulnhub.com/>

containers (**Fig. 1**). Even services provided from Linux distributions such as Kali Linux<sup>9</sup> are possible to be deployed in a docker container for the participants to use instead of a virtual machine. Towards this direction, some of the existing cybersecurity exercises that include DVWA<sup>10</sup> or Webgoat<sup>11</sup> have been released as docker containers. The main idea of our approach is that docker containers can be used for deploying multiple service instances for the participants to have their own cyberspace environment to practice with.



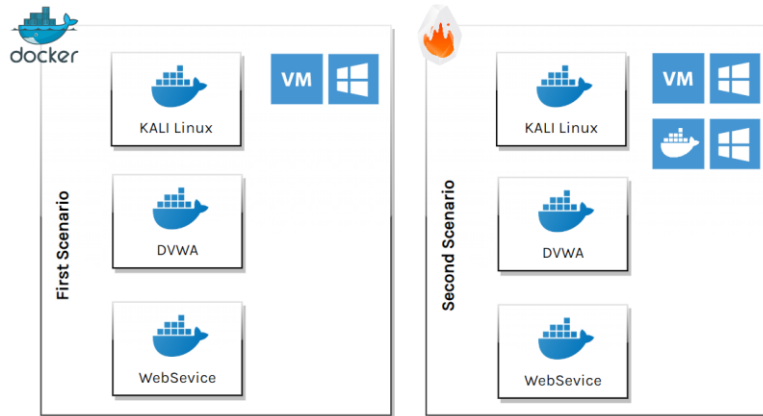
```

root@sphinx-virtual-machine: ~
root@sphinx-virtual-machine: ~
root@sphinx-virtual-machine: ~
root@sphinx-virtual-machine:~# docker ps
CONTAINER ID   IMAGE                                PORTS                NAMES
94ba526eb708   webgoat/webgoat-7.1                0.0.0.0:8088->8080/tcp  sharp_mayer
035584ca5e60   webgoat/webgoat-8.0                0.0.0.0:8085->8080/tcp  eager_buck
da2e8b88afc5   webgoat/webgoat-8.0                0.0.0.0:8084->8080/tcp  stoic_meitner
38b105fe13df   webgoat/webgoat-8.0                0.0.0.0:8083->8080/tcp  elated_wright
05daf73f6465   webgoat/webgoat-8.0                0.0.0.0:8082->8080/tcp  zealous_williams
610a6f562c86   webgoat/webgoat-8.0                0.0.0.0:8081->8080/tcp  lucid_elbakyan
eb3102cbe45e   webgoat/webgoat-8.0                0.0.0.0:8080->8080/tcp  nifty_fermi

```

**Fig. 2.** Webgoat instances running as different docker containers

Despite their benefits, containers include a few security issues, mainly deriving from the fact that they share access to a single host, meaning that any potential malicious code could get full access and take over the host system. Such incidents of escalating the privileges from a docker container are not addressed in the content of this research paper.



**Fig. 3.** Dockers and Ignite Firecracker

On the other hand, containers are easier to manage than virtual machines making it possible to create a network topology that includes more software components to properly initiate the exercises, requiring less deployment and integration effort. Despite

<sup>9</sup> <https://kali.org/>

<sup>10</sup> <https://hub.docker.com/r/vulnerables/web-dvwa/>

<sup>11</sup> <https://hub.docker.com/r/webgoat/webgoat-8.0/>

the security concerns, new containerization solutions are on the process of mitigating such threats. For example, Firecracker and more particularly Ignite Firecracker<sup>12</sup>, is an existing solution providing kernel isolation running Kernel-based Virtual Machine (KVM), while including less overhead.

As shown in **Fig. 3**, docker containers are possible to be deployed as nested containers using a specific flag for running the docker image (--privileged). Another option we investigate is the case where dockers inside a docker are deployed using the same or different docker daemon when required. Such options come with the risk of triggering inconsistencies on the processed data or creating unstable environments. Therefore, the solution of using Firecracker is more applicable providing strong isolation. Taking the above into consideration, it is important to conduct a performance evaluation and deploy test cases to understand and discover any potential security or performance issues. Using either the Dockers inside a Docker or MicroVMs with Firecracker, it is possible to deploy multiple instances for the participants to exercise, giving them the opportunity to interact with their own isolated cyberspace. The isolation capabilities along with the performance evaluation, benefits and drawbacks of each approach are presented in detail in the next section.

### 3.1 Evaluation of Popular Virtualization and Containerization Techniques

The purpose of this evaluation was to discover the performance capabilities and measure the total overhead of each of the existing techniques and to deploy various approaches for analyzing the benefits and drawbacks. In **Table 1** the compatibility capabilities and the option to deploy a system or service inside a service are presented.

**Table 1.** Capabilities for executing containerization and virtualization techniques

	KVM	Docker	Docker Compose	Firecracker	W7-10
KVM	✓	✓	✓	✓	✓
DinD	✓	✓	✓		
Docker	✓				
Firecracker	✓	✓	✓		
RancherVM	✓	✓	✓		✓ (W7)

For example, it is possible to execute a KVM inside a KVM, a docker inside a KVM, the ability to run multi-container Docker applications (docker compose) inside a KVM, running Firecracker inside a KVM and finally to test deployment options for Windows services. The above cases investigate the current deployment possibilities in response to RQ1 mentioned in Section 1 and the benefits as well as the challenges and drawbacks are described further below (**Table 1**, **Table 2**). In our tests, we discovered quite a few compatibility issues regarding Windows hosts, and we also included RancherVM in our tests as another solution for creating virtual images with less overhead and

<sup>12</sup> <https://github.com/weaveworks/ignite/>

successfully run Windows 7 machines using KVM. Our efforts for creating a Windows 10 machine for using in RancherVM was not successful and might require more effort to proceed with this approach. The approach of using RancherVM is included in **Table 2**, however we excluded the results from the evaluation considering the deployment issues we had for executing properly the evaluation tests for RancherVM (we could not deploy Windows10 hosts). An approximately summary of the performed tests is presented in terms of the total overhead, compatibility, performance, isolation capabilities and scalability per approach (**Table 2**) derived from the extracted metrics described below. The main benefits are in terms of scalability from using docker containers or firecracker and the color highlights and describes the benefits and drawbacks for each of the selected approaches. For conducting the evaluation tests, a native Linux system was used (Fedora Workstation 32) and a computer system that contained an i7-9750H CPU with 24GB DDR4 RAM memory and 1TB NVME-SSD hard disk.

**Table 2.** Summary matrix for benefits and drawbacks for each of the approaches

Benefits	KVM	Docker	Firecracker	Rancer.VM	Indices	
Less Overhead					0-20%	
Compatibility					20-40%	
Performance					40-60%	
Isolation					60-80%	
Scalability					80-100%	

In all our tests we ensured that all the other applications were closed, and no additional overhead was added except for the main system services. For the evaluation tests of the Linux hosts/services we used Sysbench<sup>13</sup> for the memory tests and Stress-ng<sup>14</sup> for testing the Control Process Unit (CPU) and collecting disk cache input/output (I/O) benchmarks. For the Windows system hosts we used Novabench<sup>15</sup>. The details of the system tests are presented in **Fig. 4** considering the following benchmarks:

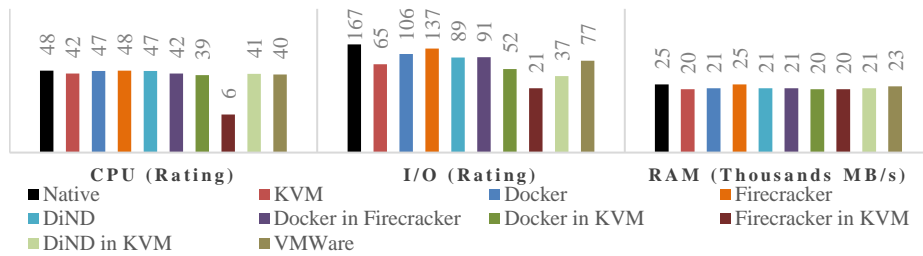
1. **CPU:** CPU performance tests using the Stress-ng for each different technology. Rating is considered as the number of iterations of the CPU stressor during the run for 20seconds.
2. **I/O – Hard disk:** Performance test using Stress-ng related to the disk's cache measuring the input/output operations per second. Rating is considered as the number of iterations of the disk cache stressor during the run for 20seconds.
3. **RAM memory:** Effective RAM performance by calculating the writing speed (Mega Bytes per second – MB/s).

<sup>13</sup> <https://github.com/akopytov/sysbench/>

<sup>14</sup> <https://wiki.ubuntu.com/Kernel/Reference/stress-ng/>

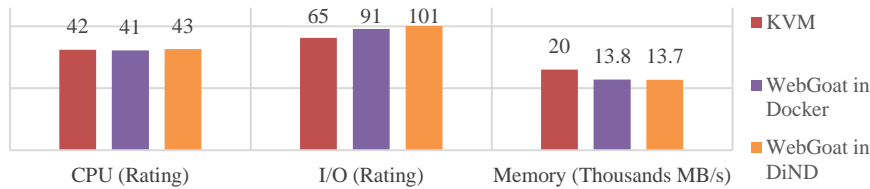
<sup>15</sup> <https://novabench.com/>

The evaluation metrics of course depend on the main system resources and our purpose was to compare the difference between the used technologies. The results from the performance evaluation and benchmarks are presented in **Fig. 4**. Taken the above into consideration the results from the performance evaluation present that docker containers maintain low overhead, mainly in terms of I/O – disk cache writing and reading speeds (**Fig. 4**) in response to RQ2 (Section 1). Furthermore, we have also investigated the total overhead in terms of both the used hard disk space and memory size for deploying the vulnerable systems or services.



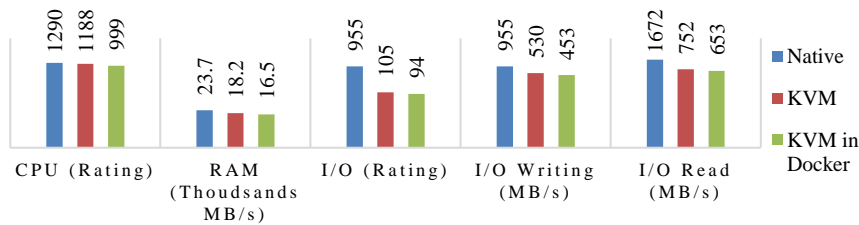
**Fig. 4.** Performance evaluation for the selected approaches

Results from the performance evaluation for WebGoat (a popular vulnerable web application for using in cybersecurity exercises), running in a Docker container instead of a KVM, are presented in **Fig. 5**. It is important to mention that the I/O hard disk latency can significantly affect the total performance.



**Fig. 5.** Performance evaluation for WebGoat

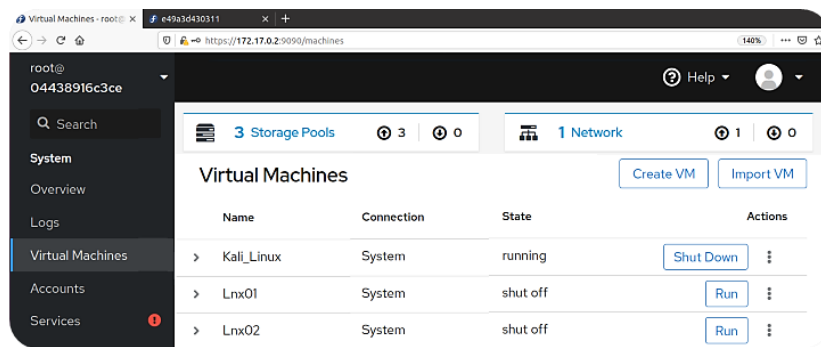
After deploying 10 different docker containers of WebGoat, it was concluded that only 931MB of the system's memory was used (398MB for deploying all the containers) instead of 12GB system memory that WebGoat required running as a virtual machine.



**Fig. 6.** Performance evaluation of Windows Hosts

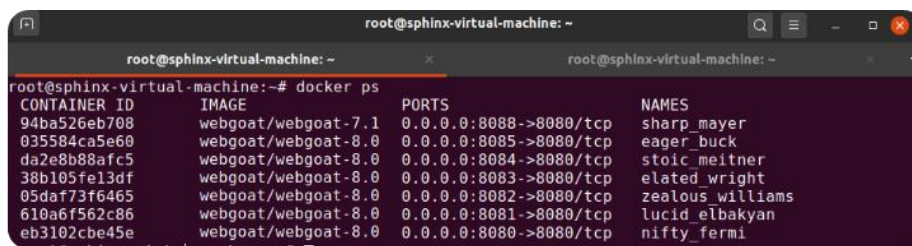


The total disk space that WebGoat required was 1.2GB, while the disk space required for deploying it as a docker was 533MB for the docker image and 398MB for each container. The deployed containers required no additional disk space after deploying the first container unless further changes to the container files are applied. Therefore, the total disk space required for deploying the services is significantly reduced. Furthermore, every docker container has a separate IP and therefore each participant is able to conduct an isolated and independent assessment to the potential vulnerable service or system. In **Fig. 6** the results from the results from the performance evaluation for Windows hosts are presented, using KVM and also the deployment of a windows hosts running on KVM in a docker container.



**Fig. 7.** KVM running in two different docker containers

As presented in **Fig. 7** each one of the deployed containers are managing KVM and include 3 already deployed virtual machines. The drawbacks from the deployments using containerization include various security risks that could allow the participants to take over the host Windows hosts currently cannot be deployed using containers, but only using KVM or similar virtualization technologies. Therefore, the total overhead in terms of disk space, RAM and CPU is difficult to be reduced when it is required to deploy a large number of windows hosts.



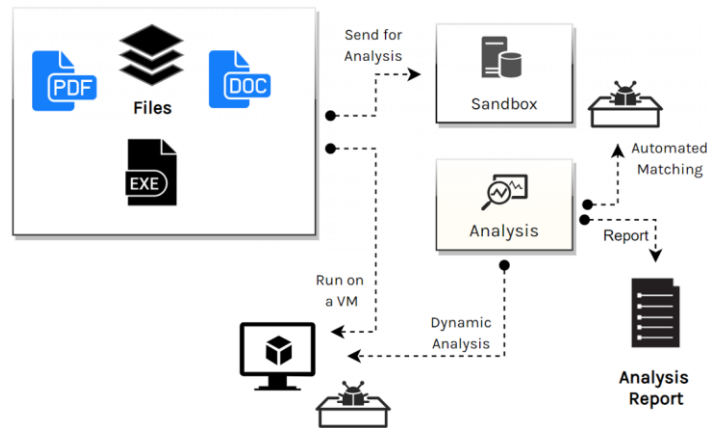
**Fig. 8.** The running docker container that include KVM and docker in a docker capabilities

The overhead for the docker container that runs the KVM service is affordable in comparison with the direct KVM deployment. A solution for the compatibility issues

of Windows hosts is to use Image2Docker<sup>16</sup> to containerize some of the workloads migrating Windows apps out of virtual machines. However, such approaches are not considered in this research paper. In summary and responding to RQ1, RQ2 it seems that both virtualization technologies and containerization technologies hold both benefits and drawback as presented in this section. However, scalability capabilities which docker containers have makes the choice for our deployment more appropriate. However, the usage of KVM is not excluded but we decided to include KVM inside a docker container for creating unique cyberspaces which are running on different containers that include KVM mostly for running Windows hosts.

### 3.2 Sandboxing for Monitoring the Participants' Actions

A sandbox, in general terms, is a testing environment which allows the validation of code, services, or software components before migrating to the production environment. In malware analysis it is important to dynamically execute auditing and monitoring processes in the virtual system Sandboxing is frequently used in cybersecurity to perform deep analysis of evasive and unknown threats. The hidden behavior of the potential malware is revealed using automated dynamic analysis or by testing the code manually. **Fig. 9** represents the process of the existing security solutions for conducting dynamic malware analysis. Malware could be difficult to detect using signature-based security solutions. Therefore, for conducting dynamic malware analysis, approaches such as Cuckoo sandbox and virtualization techniques such as KVM, VMWare<sup>17</sup> or Virtualbox<sup>18</sup> are used [21, 22].



**Fig. 9.** Existing sandbox approaches

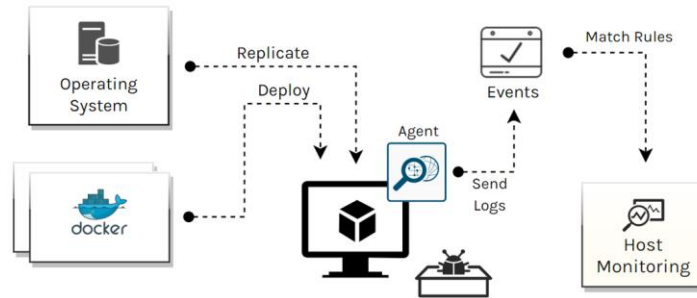
The process includes files that are sent for malware analysis to a sandbox which initiates a virtual machine for executing the file. After the execution of the file from the sandbox,

<sup>16</sup> <https://github.com/docker-archive/communitytools-image2docker-win>

<sup>17</sup> <https://vmware.com/>

<sup>18</sup> <https://virtualbox.org/>

screenshots are generated accordingly, and the system shuts down in case of malware infection. While the procedure is dynamic, the results and reports are static, solely focusing on the potential infected file. Therefore, such approaches do not include vulnerability assessments in cases where a vulnerable service is deployed that might not be malicious, however the deployed service could intentionally open specific vulnerabilities in the system (e.g. deploying an outdated apache server).



**Fig. 10.** Dynamic and continuous system auditing using sandboxing

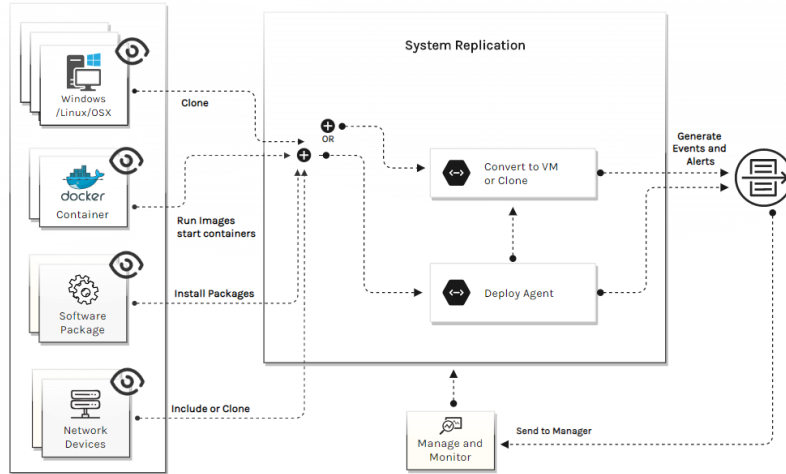
**Fig. 10** presents the possibility of conducting security auditing in systems not only for malware analysis but for overall monitoring the behavior of sandboxed systems. In our case we used Wazuh<sup>19</sup>, a host-based intrusion detection system (HIDS) for combining anomaly and signature-based technologies to detect intrusions, potential threats and behavioral anomalies triggered by the security events generated from the participants of the cybersecurity exercise. Our intention was to further extend the potential of dynamic analysis, using sandboxing to conduct security and auditing tests including procedures such as file integrity monitoring, vulnerability detection, regulatory compliance, among others.

#### 4 Towards a new model for Cyber Range deployment

Containers, as discussed above, present a lot of benefits and new technologies such as Firecracker extend the possibilities for deploying systems or services requiring less effort and inducing less overhead. The performance evaluation presented in this paper supports this fact; however, the tests were not conducted in a stressful or overloaded network environment to provide more accurate metrics regarding the system responses. Security aspects and isolation capabilities should be tested as well to better define the security posture of the proposed deployments. RancherVM was not completely tested since some deployment issues were present which would result in having additional overhead and thus it was excluded from the evaluation metrics. However, admittedly, the existing approaches could be revised to include more options and capabilities during hands-on practices. Not only the performance issues and deployment options are mature enough, but the cybersecurity exercises could extend further to more reactive security

<sup>19</sup> <https://wazuh.com/>

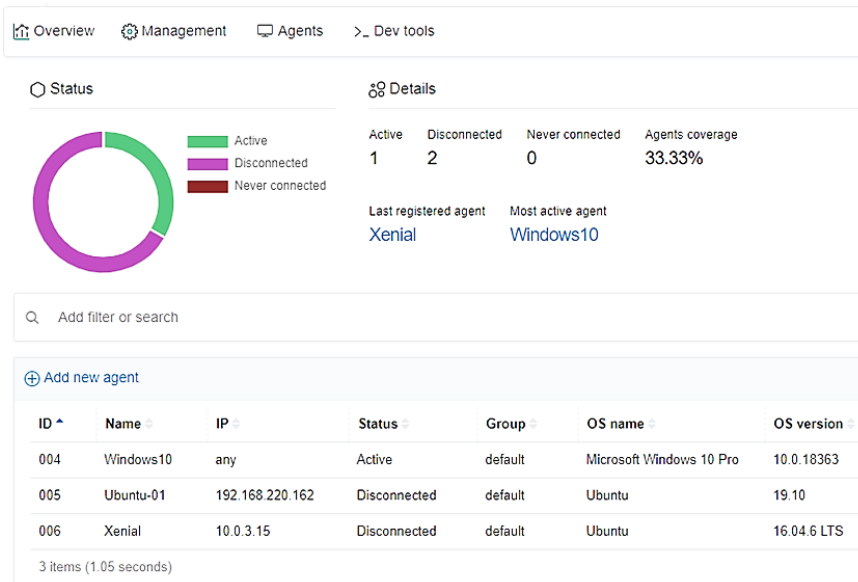
scenarios that include incident response and blue teaming. Furthermore, it is easier to deploy existing infrastructures and network topologies using both virtualization and containerization.



**Fig. 11.** A Cyber Range deployment for educational purposes

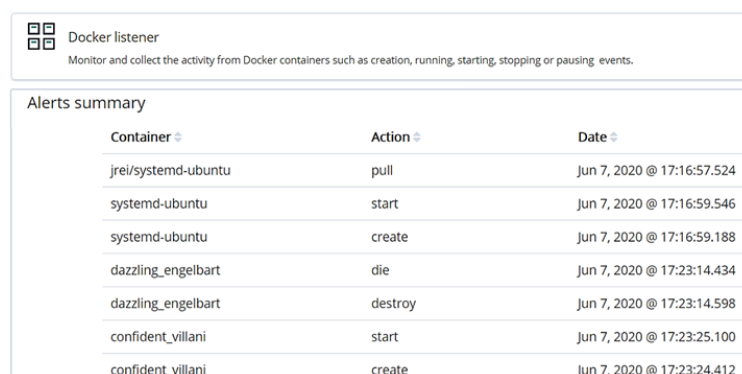
The benefits and drawbacks for each approach were presented in Section 3 and educators could use the options that fit better to their preferences. For example, the option to use Dockers inside a Docker could be easily deployed requiring less effort as it requires no further configurations and could be easily deployed in seconds. Not only this but the required hard disk size is reduced, and the facilitator could easily deploy more than one services or operating systems running in a docker container. A significant benefit from using containers instead of a virtual machine is the low overhead induced for the RAM size since virtual machines require a different kernel which requires a significant amount of memory (about 1GB for a modern Ubuntu distribution). Indeed, there are solutions which might reduce the total overhead even more, but currently the easier approach for mitigating such issues is using containerization. An approach towards this direction is presented in **Fig. 11**, presenting the possibility to replicate or sandbox entire systems deployed for educational and learning purposes.

An important aspect of such approaches is that multiple complex systems and services could interact. Therefore, we present the case where the systems are interacting with each other and participants are invited to conduct security tests or red team assessments on a cyberspace instance that includes multiple components. The network topology is automatically deployed along with the used ports providing easier deployment. The agents are already deployed as presented in **Fig. 12**, reducing the required effort for the total deployment. Finally, as mentioned in Section 3.2 the monitoring process is used for collecting the progress from each participant.



**Fig. 12.** Monitored Virtual Systems and Docker containers

As a result, the standard approaches for using CtF challenges as an assessment tool could be extended since we are able to actually monitor the participants' actions and trigger events related to security rulesets, policies or to create custom rules that matches to the offensive actions. Being able to monitor the deployed assets the exercises could include more interactive elements, enabling attack and defense scenarios extending exercises focused also on blue teaming and incident response. The monitored assets could include docker containers, services or virtual systems (**Fig. 12**, **Fig. 13**). Therefore, the total required effort decreases providing us the opportunity to deploy more complex security scenarios for conducting cybersecurity exercises.



**Fig. 13.** Capability to monitor specific docker containers

In this research paper we tested the capabilities to host and deploy the proposed approach and we successfully deployed the manager for monitoring, Webgoat and DVWA for having a hands-on lab ready for replication. The Dockerfile retrieves and deploy all the required images for the cybersecurity exercises. The APIs for executing the security scenarios and to enhance automation for deploying labs according to specific learning goals is still under research.

## **5 Conclusions and Future Work**

This paper discusses the potential benefits of using containerization instead of virtualization technologies to deploy cyberspaces for maintaining cybersecurity exercises. In response to our research questions mentioned in Section 1, we discovered the various features that containerization and MicroVMs provide, in contrary to virtualization technologies (RQ1). More specifically, docker containers include a lot of benefits and more specifically the reduced overhead and of the required system resources, however specific security issues apply. In response to RQ2, we concluded that by using containerization techniques or MicroVMs, the overall overhead is reduced in comparison with the traditional virtualization technologies. More specifically, a significant reduction in terms of the used memory and amount of disk was observed, among other performance benefits. Towards this direction, we created a docker image that contained multiple docker containers for the facilitators or educators to deploy Cyber Ranges. The results confirm that the total overhead is decreased, and that the total management is easier for creating and deploying cybersecurity hands-on labs.

Future work includes the creation or alignment of the rulesets that will apply for monitoring the participants' progress by collecting security events that triggered from their offensive tasks. Furthermore, specific cybersecurity exercises need to be deployed for further testing the appropriateness of our proposal. Extended research will be carried on deploying specific Common Vulnerabilities and Exposures (CVEs) using docker containers. Finally, the connection to the National Initiative for Cybersecurity Education (NICE) from NIST is in scope of our future research as well as the revision of the existing cybersecurity exercises to align with our approach. Towards this direction we intend to further investigate the existing taxonomies that might help identifying the learning impact during the exercises.

## **Acknowledgments**

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 have not participated in the elaboration of this research paper.

## References

1. Childers, N., Boe, B., Cavallaro, L., Cavedon, L., Cova, M., Egele, M., Vigna, G.: Organizing large scale hacking competitions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 6201 LNCS, 132–152 (2010).
2. Irvine, C.E., Michael, F., Khosalim, J.: *Labtainers: A Framework for Parameterized Cybersecurity Labs Using Containers*. (2017).
3. Schreuders, Z.C., Shaw, T., Shan-A-Khuda, M., Ravichandran, G., Keighley, J., Or-dean, 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).
4. Hay, B., Dodge, R., Nance, K.: Using virtualization to create and deploy computer security lab exercises. *IFIP International Federation for Information Processing*. 278, 621–635 (2008).
5. B, K.T., Abujelala, M., Rajavenkatanarayanan, A.: Interfaces for Personalized Robot-Assisted Training. 88–98 (2018).
6. Furnell, S., Fischer, P., Finch, A.: Can't get the staff? The growing need for cyber-security skills. *Computer Fraud and Security*. 2017, 5–10 (2017).
7. Burley, D.L.: Special section: Cybersecurity education, Part 2. *ACM Inroads*. 6, 58–59 (2015).
8. Baldassarre, M.T., Barletta, V.S., Caivano, D., Raguseo, D., Scalera, M.: Teaching cyber security: The hack-space integrated model. *CEUR Workshop Proceedings*. 2315, (2019).
9. Zimmerman, C.: *Cybersecurity Operations Center*. (2014).
10. Debatty, T., Mees, W.: Building a Cyber Range for training CyberDefense Situation Awareness. *2019 International Conference on Military Communications and Information Systems, ICMCIS 2019*. 1–6 (2019).
11. Beltran, M., Calvo, M., Gonzalez, S.: Experiences using capture the flag competitions to introduce gamification in undergraduate computer security labs. *Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018*. 574–579 (2018).
12. Thompson, M.F., Irvine, C.E.: Individualizing Cybersecurity Lab Exercises with Labtainers. *IEEE Security and Privacy*. 16, 91–95 (2018).
13. AlSalamah, A.K., Cámara, J.M.S., Kelly, S.: Applying virtualization and containerization techniques in cybersecurity education. *Proceedings of the 34th Information Systems Education Conference, ISECON 2018*. 1–14 (2018).
14. Perrone, G., Romano, S.P.: The docker security playground: A hands-on approach to the study of network security. *2017 Principles, Systems and Applications of IP Telecommunications, IPTComm 2017*. 2017-Septe, 1–8 (2017).
15. Yin, Y., Shao, Y., Wang, X., Su, Q.: A Flexible Cyber Security Experimentation Plat-form Architecture Based on Docker. *Proceedings - Companion of the 19th IEEE International Conference on Software Quality, Reliability and Security, QRS-C 2019*. 413–420 (2019).
16. Du, W.: SEED: Hands-on lab exercises for computer security education. In: *IEEE Security and Privacy*. pp. 70–73 (2011).
17. Baillie, C., Standen, M., Schwartz, J., Docking, M., Bowman, D., Kim, J.: *CybORG: An Autonomous Cyber Operations Research Gym*. (2020).
18. Costa, G., Russo, E., Armando, A.: Automating the Generation of Cyber Range Virtual Scenarios with VSDL. (2020).

19. Chaskos, E.C.: Cyber-security training: A comparative analysis of cyber- ranges and emerging trends. 78 (2019).
20. Vykopal, J., Vizvary, M., Oslejsek, R., Celeda, P., Tovarnak, D.: Lessons learned from complex hands-on defence exercises in a cyber range. Proceedings - Frontiers in Education Conference, FIE. 2017-October, 1–8 (2017).
21. Jamalpur, S., Navya, Y.S., Raja, P., Tagore, G., Rao, G.R.K.: Dynamic Malware Analysis Using Cuckoo Sandbox. Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018. 1056–1060 (2018).
22. Keahey, K., Doering, K., Foster, I.: From sandbox to playground: Dynamic virtual environments in the grid. Proceedings - IEEE/ACM International Workshop on Grid Computing. 3, 34–42 (2004).