CYBERSPACE SANDBOXING FOR CYBERSECURITY EDUCATION AND LEARNING

Seelam Namrata[1], Deeptansu Jena[2], Indu Khatri[3]
[1,2] B.tech Student, [3]Assistant Professor
Department of Computer Science Engineering
Bhagwan Mahaveer College of Engineering and Management, Sonipat

Abstract: Creating a suitable digital environment for performing cybersecurity exercises can be difficult and time-consuming, and it usually necessitates a significant amount of effort and system resources. More configuration, as well as technical expertise, is usually necessary for implementing vulnerable web services and setting up laboratories for hands-on cybersecurity exercises. Containerization strategies and solutions have less overhead and can be utilized to improve existing systems instead of virtualization techniques. In order for cybersecurity exercises to be realistic, existing systems or services must be sandboxed or replicated. We investigated strategies related to containerization or MicroVMs that have lower overhead than traditional virtualization techniques in order to give meaningful and comparable outcomes from the deployment of scalable solutions, as well as their benefits and cons. Finally, we proposed a use case for delivering cybersecurity exercises that involves minimal work and only moderate system resources, as well as a method for monitoring participants' progress using a host-based intrusion system.

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

1. INTRODUCTION

Designing and delivering effective cybersecurity labs necessitates the use of a variety of technologies, as well as a significant amount of effort for the deployment of effective cybersecurity exercises. For computer security students to benefit from hands-on experiences, a wide range of security technologies must be used, making the process of correctly creating and deploying the exercises complicated and time-consuming. Virtualization technologies make it possible to host numerous machines on a single system, reducing the amount of time and resources necessary for deployment and boosting the instructor's capacity to build complicated Scenarios for educational reasons. Our goal is to develop a modular and portable solution that does not require any current infrastructure, as well as to deploy various systems for security testing that involves complicated procedures like adversary emulation and incident response. Existing cybersecurity exercises are frequently conducted utilizing virtualization, which limits learning processes while obviating the need for major interactions between deployed services (e.g. interactions between Intrusion detection systems, adversaries, and the usage of Elastic Search). Not only that, but virtual machines come with a lot of overhead and a large demand on system resources for implementing the exercises' related services. Security Operations Center (SOC) teams, for example, are designed to provide high-quality IT-security services by employing systems that actively detect and respond to possible threats and assaults. Finally, the learning outcomes of hands-on practices must adhere to curriculum guidelines or frameworks that address the collaborative actions necessary in cybersecurity between industry, government, and academic institutions.

Docker containers, Linux Containers (LXC), MicroVMs, RancherVM, and other choices for delivering and running Kernel-based Virtual Machine (KVM) or docker containers inside a docker are examples of modern technologies for providing services or operating systems. Current operating systems, particularly Linux variants, improve the ability to deploy in a portable and flexible manner. As a result, current exercises and tools may be more easily deployed and managed.

This study aims to examine state-of-the-art ways for deploying cyber security exercises, taking into account portability, flexibility, and the ability to provide easy-deployment while lowering total overhead and system resource requirements. Our goal is to implement, assess, and research the best methods for leveraging such technologies to keep cybersecurity exercises and hands-on labs running smoothly while requiring less deployment time. As a result, the key research issues that this work tries to answer are as follows:

RQ1: What are the advantages, disadvantages, and benefits of the various virtualization or containerization technologies for creating and deploying cybersecurity exercises?
RQ2: What are the best strategies for conducting sophisticated cybersecurity exercises with minimal resource overhead and increased compatibility?

RQ1 and RQ2 will examine existing deployment choices for cybersecurity exercises employing sandboxing, as well as explore the present capabilities of containerization and virtualization technologies. To this end, we conducted an in-depth analysis and performance review of the most used technologies, as well as sample activities to learn about the merits and downsides of each strategy. The following is the layout of the research paper: Section 3 examines common virtualization technologies and containerization methodologies, demonstrating the potential to use a sandbox as the primary learning environment. Part 4 discusses the use of a sandbox as a possible option for sustaining complicated Cyber Ranges, and section 5 discusses future action points.

2. RELATED WORK

Using LXC or Docker containers instead of virtual machines has been investigated during the last few years . Irvine et al., for example, published a paradigm for parameterizing cybersecurity laboratories with containers. The main advantages of using containers instead of virtual machines are their higher performance, which allows for easier deployment of a large number of systems and services while using fewer resources. AlSalamah et al., for example, examine how containerization approaches could open up new options, emphasising the distinction between virtual machines and containers. They look at the advantages of containers in terms of configuration, networking, and performance, as well as their flexibility in terms of deploying a large number of services. Similarly, research has been done on the construction of architectures and toolsets for offering learning cyberspaces linked to network security and producing hands-on lab exercises. The major advantages of adopting dockers rather than virtualization solutions are also noted in their research.

ENISA's Computer Security and Incident Response Team (CSIRT) has published and delivered training material that is continually updated by new exercise scenarios incorporating toolsets and virtual images to facilitate hands-on training sessions, starting in 2008.

Workstations, firewalls, and servers, among other things, are crucial aspects of enterprise networks for providing a high-fidelity training environment. Instead, CtF (Capture the Flag) exercises rarely feature deployment choices that involve more complicated infrastructures and network topologies, and they are usually associated with cyber ranges, where more complex topologies are given. As a result, cybersecurity exercise deployment options are presently being changed to use containerization and virtualization technologies to extend and deliver more interactive cybersecurity learning settings.

3. VIRTUALIZATION TECHNOLOGIES AND SANDBOXING

For testing purposes, virtualization technologies are widely used to create and install insecure virtual systems. HackTheBox, TryHackMe, and the susceptible images provided on VulnHub, an open repository providing hands-on lab cybersecurity exercises, are also popular methods.

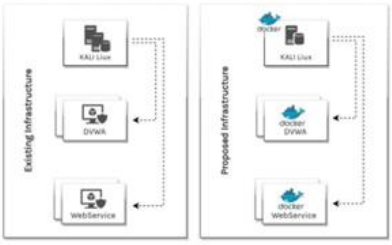

Fig. 1. Dockerization of existing services and vulnerable systems

To assist hands-on training sessions, SEED labs and ENISA CSIRT produced training material incorporating cybersecurity exercise situations in the form of Virtual Images. As a result, certain cybersecurity circumstances can be tweaked to reduce overall overhead. Existing services, for example, can be updated and deployed as Docker containers (Fig. 1). Even Linux distribution services, such as Kali Linux9, can be placed in a Docker container for the participants to utilize instead of a virtual machine. The main idea behind our method is that Docker containers may be used to deploy several service instances, allowing users to experiment in their own virtual environment.

Fig. 2. Webgoat instances running as different docker containers

Despite their advantages, containers have several security vulnerabilities, mostly due to the fact that they share access to a single host, allowing any malicious code to gain complete access and take control of the host system.

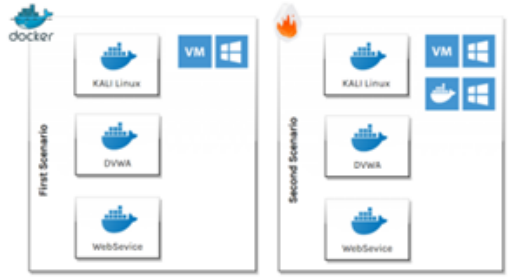

Fig. 3. Dockers and Ignite Firecracker

Containers, on the other hand, are easier to handle than virtual machines, allowing for the creation of a network topology with more software components to adequately begin the exercises with less deployment and integration effort.Firecracker, more specifically Ignite Firecracker, is an existing solution that provides kernel isolation while running a Kernel-based Virtual Machine (KVM) with reduced overhead.

Docker containers can be deployed as nested containers using a special flag for running the docker image, as seen in Figure 3. Such options run the risk of causing inconsistencies in the processed data or resulting in unstable environments. As a result, utilizing Firecracker is a better approach for ensuring tight isolation.

In light of the foregoing, it's critical to conduct a performance review and deploy test cases in order to identify and resolve any potential security or performance issues. It is feasible to create numerous instances for the participants to exercise using either Dockers inside a Docker or MicroVMs with Firecracker, providing them the opportunity to engage with their own isolated cyberspace. The next part delves into the isolation capabilities, as well as the performance evaluation, benefits, and downsides of each strategy.

4. VIRTUALIZATION AND CONTAINERIZATION TECHNIQUES: A REVIEW

The goal of this assessment was to determine the performance capabilities and total overhead of each available technique, as well as to test several approaches for appraising the benefits and drawbacks. Table 1 shows the compatibility possibilities as well as the opportunity to deploy a system or service within a service.

Table 1. Capabilities for executing containerization and virtualization techniques

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

The following scenarios look at the existing deployment options in response to RQ1 in Section 1, and the benefits, as well as the obstacles and drawbacks, are detailed below (Table 1, Table 2). We uncovered a number of compatibility concerns with Windows hosts during our tests, and we also introduced RancherVM as another option for producing virtual images with minimal overhead and successfully running Windows 7 workstations with KVM. Our attempts to create a Windows 10 machine for use with RancherVM were unsuccessful, and it is possible that this strategy will take more effort in the future. Table 2 shows the RancherVM strategy, but the findings were left out of the evaluation due to deployment concerns with RancherVM (we could not deploy Windows10 hosts). The total overhead, compatibility, performance, isolation capabilities, and scalability per strategy (Table 2) are approximate summaries of the performed tests obtained from the extracted metrics given below. The major advantages of using docker containers or firecracker in terms of scalability, and the colour emphasises and illustrates the advantages and disadvantages of each strategy. A native Linux system (Fedora Workstation 32) and a computer system with an i7-9750H CPU, 24GB DDR4 RAM, and a 1TB NVME-SSD hard disc were used to conduct the evaluation tests.

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%

We utilised Sysbench for memory tests and Stress-ng to test the Control Process Unit (CPU) and collect disc cache input/output (I/O) benchmarks for the Linux hosts/services evaluation tests. Novabench was used to test the Windows system hosts. The system tests are detailed in Fig. 4 and are based on the following benchmarks:

1. CPU: Stress-ng is used to measure CPU performance for each technology. The number of iterations of the CPU stressor during the 20-second run is used to determine the rating.
2. I/O – Hard drive: Stress-ng performance test of the disc cache, evaluating the number of input/output operations per second. The number of iterations of the disc cache stressor during the 20-second run is used to determine the rating.
3. RAM memory: Calculate the writing speed (Mega Bytes per Second – MB/s) to determine effective RAM performance.

Figure 4 depicts the outcomes of the performance evaluation and benchmarks. In light of the foregoing, the results of the performance evaluation show that docker containers have minimal overhead, particularly in terms of I/O – disc cache writing and reading rates (Fig. 4), in response to RQ2 (Section 1).

Fig. 4. Performance evaluation for the selected approaches

Figure 5 shows the results of a performance evaluation for WebGoat operating in a Docker container rather than a KVM.

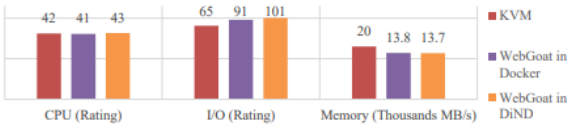

Fig. 5. Performance evaluation for WebGoat

After deploying 10 distinct WebGoat docker containers, it was discovered that only 931MB of the system's RAM was consumed (398MB for deploying all the containers), less than the 12GB required by WebGoat when running as a virtual machine.

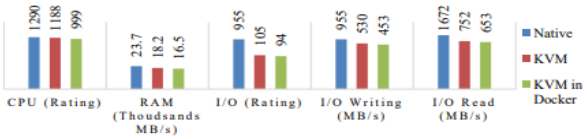

Fig. 6. Performance evaluation of Windows Hosts

WebGoat required 1.2GB of total disc space, with 533MB for the docker image and 398MB for each container when deployed as a docker. As a result, the total amount of disc space required to deploy the services is drastically decreased. Additionally, because each docker container has its own IP, each participant can undertake an isolated and independent assessment of the potentially susceptible service or system. The results of the performance evaluation for Windows hosts utilising KVM, as well as the deployment of Windows hosts running on KVM in a Docker container, are shown in Figure 6.

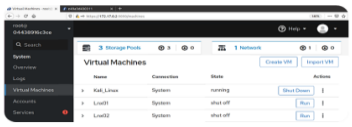

Fig. 7. KVM running in two different docker containers

Each of the deployed containers, as shown in Fig. 7, manages KVM and contains three virtual machines that have already been deployed. The disadvantages of containerized deployments include different security vulnerabilities that could allow participants to take control of the host As a result, when deploying a large number of Windows hosts, the total overhead in terms of disc space, RAM, and CPU is difficult to lower.

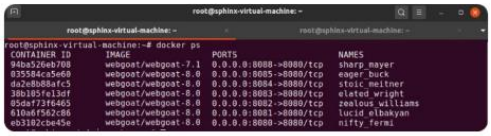

Fig. 8. The running docker container that include KVM and docker in a docker capabilities

In comparison to a straight KVM deployment, the overhead for the docker container that executes the KVM service is reasonable. Use Image2Docker16 to containerize part of the workloads transferring Windows apps out of virtual machines to solve the compatibility difficulties of Windows hosts. In response to RQ1 and RQ2, it appears that both virtualization and containerization technologies have benefits and drawbacks, as described in this section. Although the use of KVM is not prohibited, we choose to incorporate it inside a docker container in order to create unique cyberspaces that operate on separate containers that feature KVM, which is primarily used to run Windows hosts.

5. SANDBOXING FOR MONITORING THE PARTICIPANTS' ACTIONS

In general, a sandbox is a testing environment that allows code, services, or software components to be validated before being deployed in the production environment. Sandboxing is a popular cybersecurity technique for performing in-depth investigation of elusive and unexpected threats. Using automatic dynamic analysis or manually testing the code, the hidden behavior of the potential malware is exposed. Figure 9 depicts the method of conducting dynamic malware analysis using existing security technologies. Signature-based security solutions may have difficulty detecting malware. As a result, technologies like Cuckoo sandbox and virtualization techniques like KVM, VMWare17, or Virtualbox18 are employed to conduct dynamic malware research.

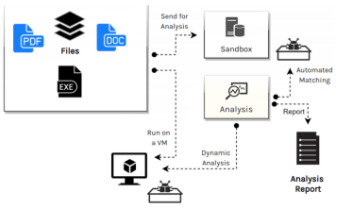

Fig. 9. Existing sandbox approaches

Files are submitted to a sandbox for malware analysis, which creates a virtual machine in which the file can be executed. In the event of malware infection, screenshots are generated after the sandbox file is executed, and the system is shut down. The technique is dynamic, but the results and reports are static, focused primarily on the possible infected file. As a result, vulnerability evaluations are not included in such techniques in circumstances where a susceptible service is launched that may or may not be malevolent, but the deployed service may open certain vulnerabilities in the system intentionally (e.g. deploying an outdated apache server).

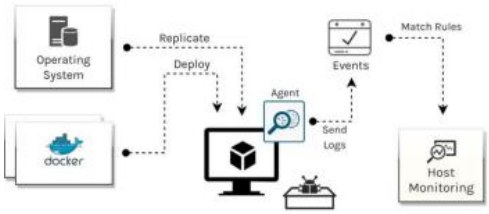

Fig. 10. Dynamic and continuous system auditing using sandboxing

Figure 10 shows how security auditing in systems can be used not only for malware research, but also for overall monitoring of sandboxed systems' activity. We employed Wazuh, a host-based intrusion detection system (HIDS) that combines anomaly and signature-based technologies to detect intrusions, possible threats, and behavioral anomalies prompted by the cybersecurity exercise participants' security events. Our goal was to use sandboxing to undertake security and auditing checks, including file integrity monitoring, vulnerability identification, and regulatory compliance, among other procedures.

6. TOWARDS A NEW CYBER RANGE DEPLOYMENT MODEL

As previously mentioned, containers offer numerous advantages, and new technologies like Firecracker expand the possibilities for deploying systems or services with less work and overhead. This is supported by the results of the performance evaluation reported in this research; however, the tests were not done in a demanding or overloaded network environment to offer more accurate metrics on system responses. To better describe the security posture of the proposed deployments, security aspects and isolation capabilities should be examined, as well. RancherVM was not fully tested due to various deployment issues that would result in additional overhead, hence it was left out of the assessment metrics. Not only are the performance concerns and deployment alternatives mature enough, but the cybersecurity drills might be expanded to include more reactive security scenarios like incident response and blue teaming. Furthermore, both virtualization and containerization make it easier to deploy existing infrastructures and network topologies.

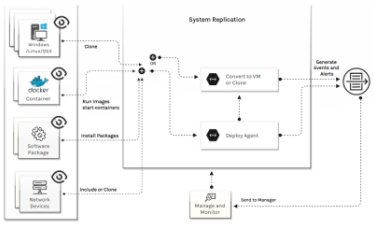

Fig. 11. A Cyber Range deployment for educational purposes

Section 3 detailed the advantages and disadvantages of each technique, allowing educators to choose the approach that best suited their needs. For example, the option to use Dockers inside a Docker may be readily installed with minimal effort because it does not require any additional configurations and can be launched in seconds. Furthermore, the required hard disc size is lowered, and the facilitator can more easily deploy many services or operating systems in a docker container. The reduced RAM overhead generated by containers instead of virtual machines is a big benefit, as virtual machines require a distinct kernel, which takes a large amount of memory (about 1GB for a modern Ubuntu distribution). There are alternatives that could cut total overhead even further, but containerization is currently the easiest way to deal with such concerns. Figure 11 depicts one technique in this area, demonstrating the ability to reproduce or sandbox entire systems used for educational and learning reasons.

Multiple complicated systems and services could interact, which is a key feature of such techniques. As a result, we propose a scenario in which systems interact and participants are asked to execute security tests or red team evaluations on a cyberspace instance with many components. The network topology and used ports are automatically installed, making deployment easier. The agents have already been deployed, as seen in Fig. 12, minimizing the total deployment effort. Finally, as described in Section 3.2, the monitoring procedure is employed to collect each participant's progress.

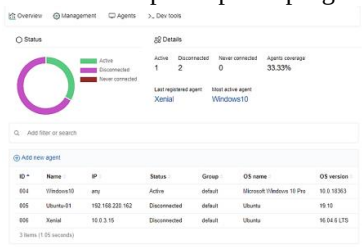

Fig. 12. Monitored Virtual Systems and Docker containers

As a result, typical ways for using CtF challenges as an assessment tool could be expanded because we can really watch the players' behaviours and trigger events linked to security ruleets, policies, or custom rules that match the offending actions. With the ability to monitor deployed assets, exercises might incorporate more interactive aspects, allowing for attack and defence scenarios, as well as blue teaming and incident response. Docker containers, services, or virtual systems might all be monitored assets (Fig. 12, Fig. 13).

Fig. 13. Capability to monitor specific docker container

We successfully implemented the manager for monitoring, Webgoat, and DVWA to have a hands-on lab ready for replication in this study report. All of the essential images for the cybersecurity exercises are retrieved and deployed via the Dockerfile.

7. CONCLUSIONS AND FUTURE WORK

The potential benefits of employing containerization instead of virtualization technologies to deploy cyberspaces for cybersecurity exercises are discussed in this study. In answer to the study questions posed in Section 1, we determined that, in contrast to virtualization technologies, containerization and MicroVMs offer a variety of benefits (RQ1). More specifically, docker containers offer a number of advantages, like decreased overhead and system resource requirements, however there are certain security concerns. We found in response to RQ2 that employing containerization techniques or MicroVMs reduces overall overhead when compared to traditional virtualization technologies. Specifically, among other performance gains, a considerable reduction in the amount of utilised RAM and disc space was noticed. We produced a docker image that contained numerous docker containers for facilitators or educators to deploy Cyber Ranges in this approach. The findings show that overall overhead is reduced, and total management for building and delivering cybersecurity hands-on laboratories is simplified.

The establishment or alignment of the rulesets that will be used to monitor the participants' progress by gathering security events produced by their aggressive tasks will be done in the future. In addition, specific cybersecurity exercises must be implemented to further test the suitability of our idea. Docker containers will be used to deploy certain Common Vulnerabilities and Exposures (CVEs). Finally, our future study will include a connection to NIST's National Initiative for Cybersecurity Education (NICE), as well as a redesign of existing cybersecurity exercises to align with our approach. We plan to look into existing taxonomies to see if they may help us identify the learning impact during the activities.

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., Rvichandran, 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 RobotAssisted Training. 88–98 (2018).

