A Container-Based Virtual Laboratory for Internet Security e-Learning

Johannes Harungguan Sianipar, Christian Willems, and Christoph Meinel
Hasso Plattner Institute, University of Potsdam, Germany
Email: firstname.lastname@hpi.de

Abstract—Tele-Lab is a platform for e-learning in Internet security with a special focus on teaching by means of hands-on experience. A virtual laboratory is implemented for the provision of training environments for practical exercises. Tele-Lab uses virtual machine (VM) technology and virtual network devices. A VM is used to represent a physical computer in an exercise scenario. While every VM needs a specific resource allocation, availability and scalability of Tele-Lab become an issue. Scaling out Tele-Lab to a public cloud is one alternative to making it more available and scalable. Public clouds are very flexible in providing resources since customers can add or reduce resources whenever needed. A second alternative is the reduction of resources needed for a single training environment, e.g., by replacing the virtualization technology in Tele-Lab with the one that uses fewer resources than full-fledged virtual machines. A container uses far fewer resources than a virtual machine. The paper at hand elaborates on the use of containers (i.e. Docker) in the training environment of a virtual machine. This work is part of the continuous improvement on Tele-Lab to make it more reliable and more scalable.

Index Terms—e-Learning, virtual laboratory, container, cybersecurity training

I. INTRODUCTION

In a learning system, students should be able to do practical hands-on-exercises. Especially for IT security subject, where the students need to practice their knowledge in a laboratory to be able to have a clear understanding and to be able to have skills in IT security. In IT Security exercises, each student need at least two interconnected machines to be able to implement an exercise scenario and check the result. One machine as a victim and the other as an attacker.

The classical approach in building IT Security laboratory is by providing a dedicated computer lab [1] [2]. Each machine is represented by a dedicated computer. Interconnection between each machine is provided by using a network device such as switch and router. The drawbacks of this classical approach are immobility, expensive to purchase and maintain, and requires a firewall to isolate the lab from outside network [3], [4]. IT Security laboratory needs to be isolated from outside network such as main university network or the Internet. This is to prevent the outside network from being endangered by the activities in the IT security laboratory. Sometime isolation for one practical environment is also needed to get a controllable environment. So besides isolating a whole laboratory from the outside network, in certain case, more granular isolation is needed for a practical environment (part of a laboratory) on a certain exercise scenario. Example: in MITM attack practical exercise, one training environment should be isolated from another. This is because of one MITM attack can affect the whole network.

In distance learning (Tele-teaching) system, each machine is represented by a Virtual Machine (VM) [3]-[5]. A VM can be provided by a hypervisor [3], a private cloud [5], or a public cloud [4]. Interconnection between VMs is provided by virtual switch. Certain VM can be accessed remotely using remote desktop by a participant to be able to execute activities based on the exercise scenario [3], [4]. Isolation is given by using VLAN and IPTables. The VLAN is used to isolate an exercise network for a participant [3]. IPTables is used to isolate the whole exercise network from the outside network. Using this distance learning system can answer the drawbacks of the classical approach.

A VM needs a fixed allocation of resources such as memory, processor and hard disk. To deliver a Man in the middle attack exercise scenario for a participant, needs 3 VMs where one VM as an attacker, and two VMs as the victims [3]. For 100 students, they must be provided with 300 VMs. Suppose one VM needs memory around 512 MB, then for 300 VMs will need 150 GB memory. To serve this number of simultaneous participants in a private cloud, needs to purchase a high specification computer hardware. In order to serve more participants, virtual laboratory must be able to scale out. [6] tried to increase scalability by integrating two Virtual Laboratories (Tele-Lab) where these two virtual laboratories physically located in different countries (Germany and Lithuania), [4] tried to increase scalability by using private cloud. [5] put the virtual laboratory in public cloud (Amazon) to make the virtual laboratory more scalable and to remove downtime during scaling hardware resources. These three approaches tried to increase scalability by providing more resources. Providing more resources means more payments. The limit is the customer budget that is available for this virtual laboratory.

To be able to serve more students like hundreds or thousands of simultaneous access, a new virtual
laboratory system must be created. This new system must use fewer resources compared to the current system. In virtualization technology, besides Virtual Machine, Container is another way to represent a machine in the context of training environment. Container is a virtualization in the application level, above the kernel. So, containers are running above the same kernel (Operating System). The Container does not install a new operating system, but used the same OS as the host. The container does not need a preliminary fixed allocation of resources. It uses name spaces and cgroup to isolate one container from the others and from the host [7]. A container is used as a host in a training environment. They can communicate between each other and act according to the practical exercise scenario. Network tools that are needed in the practical exercise scenario can be installed as needed in each container. A container uses resources just like an application in an operating system. This means that we can have container as much as available memory in a host. A container does not need resources as much as a VM. You can run container as much as you can as long as the PID and resources are available. A container can replace a VM. So, for a MITM attack exercise scenario, 100 participants need 300 containers. The number of containers that can be run on a host, is depending on how much memory is needed for an application inside the container. [8] elaborates the advantages of using a container compares to a VM.

The Research problem of this paper is on how to increase the scalability of a Virtual Laboratory (Tele-Lab) system, so the simultaneous participants can be increased significantly. There are Learning Units in Tele-Lab where every learning unit has different requirements. The proposed new system with the increasing scalability must be able to provide the requirements of every learning unit or be able to provide an alternative to replace the requirements without losing the objectives of the learning unit. Investigation must be done for every learning unit, but in this paper, we focus on the Man in the middle attack and the firewall (iptables) learning unit.

Solutions. Placing Tele-Lab in a public cloud can increase the scalability and availability, because the public cloud provider can flexibly provide the resources that is needed by the customer as long as the customer is willing to spend the money for the use of the resources. Beside it, another way to increase the scalability is by reducing the number of resources that is needed by Tele-lab. To reduce the number of resource usage, VMs should be replaced with containers.

We present a new virtual laboratory environment for Tele-lab. This new environment uses container as a virtual host in the exercise scenario. This new environment design could be implemented in private or public cloud. We use docker to provide the containers into the system. By using a container, we can isolate the virtual laboratory from the outside (internet) environment, and can also isolate one training environment from another training environment in a virtual laboratory, so participants will not interrupt each other.

Contribution. The existing virtual laboratory for IT Security e-learning such as Tele-lab uses virtual machine as a virtual host in the laboratory environment. Virtual machine needs a fixed allocation of resources such as CPU and Memory. These existing virtual laboratories have a problem with scalability. Our contribution is providing a Tele-Lab system that uses containers instead of VMs as virtual hosts inside the training environment. We investigated whether all the exercise scenario in the Tele-lab could be implemented in a container or not. By using containers we propose a system which is more scalable and can serve more participants like thousands of participants simultaneously.

II. TELE-LAB: A REMOTE VIRTUAL SECURITY LABORATORY

This paper is a part of continuing work on Tele-Lab to make it more reliable, more accessible and more scalable. The Tele-Lab platform (http://www.tele-lab.org/) was initially proposed as a stand alone system [9] were the learn platform must be installed on hard disk. Later, Tele-Lab platform was enhanced to a live DVD system, introducing virtual machines for the hands-on training [10] to make it more portable, reliable and easy to use without hard disk installation. Tele-Lab then emerged to a server system [11], [12] to integrate laboratory environment and practical exercises to e-Learning systems. This server system is accessible via Internet. In order to serve more participants, [6] tried to increase scalability by integrating two Tele-Lab servers where they physically located in different countries (Germany and Lithuania). The latest enhancement of the Tele-Lab server system is by moving to a private cloud framework to make Tele-Lab’s virtual laboratory environment more flexible, scalable and faster [5].

The Tele-Lab server basically consists of a web-based tutoring system and a training environment built on virtual machines. The tutoring system presents learning units that do not only offer information in the form of text or multimedia, but also practical exercises. Learning units of Tele-lab is described in [3]. Students perform those exercises on virtual machines on the server, which they operate via remote desktop access. Virtual machine technology allows easy deployment and recovery in case of failure. Tele-Lab uses this feature to revert the virtual machines to the original state after each usage. With the release of the current iteration of Tele-Lab, the platform was enhanced with dynamic assignment of more than one virtual machine to a single user at the same time. Those machines are connected within a virtual network providing the possibility to perform basic network attacks such as interacting with a virtual victim (e.g. Port scanning). A victim is the combination of a suitable configured virtual machine running all needed services and applications and a collection of scripts that simulate user behavior or react to the Attacker actions. They provide isolation using VLAN and IP Tables. Tele-lab have been using private cloud (Open Nebula) to provide
virtual machines. In this case, they can manage their resources as needed.

III. CONTAINER-BASED VIRTUAL LABORATORY

There are two ways to increase the scalability of a Virtual Laboratory. The first is to provide more resources and the second is to reduce the use of resources. A flexible and economic way to provide more resources is using public cloud, where resources can be provided dynamically as needed without downtime. Placing Virtual Laboratory (Tele-Lab) in the public cloud increases its scalability and flexibility, because a public cloud provider can give more resources whenever the customer needs it. Of course, this is not without limitation, as for the customer, the most influencing parameter that limits the use of resources is the ability to pay for the cost.

Replacing VMs with containers can significantly reduce the use of resources in a Virtual Laboratory. [7] and [8] describe the differences of a Container compared to a VM. A container uses much fewer resources than a VM. We use Tele-lab as the basis to implement Container-Based virtual laboratory. We enhance Tele-lab to be more scalable by replacing VMs with Containers.

Moving from VM based virtual laboratory to Container based virtual laboratory, changes several parts of the Tele-Lab system. [5] describes the architecture of Tele-Lab in a private cloud. In the existing Tele-Lab architecture, VMs are used. Replacing VMs with containers needs to create a new architecture which is a re-factored enhancement to the infrastructure presented in [5]. Basically, the architecture is almost the same as the private cloud Tele-lab, except that it uses Containers instead of VMs to represent a machine in the virtual laboratory. A host of containers is a VM in a private/public Cloud. Additional host (VM) can be provided in the same cloud provider to serve larger participants. Fig. 1 shows the architecture of Tele-Lab using containers.

As described in [3] and [5], Tele-Lab has a Front-end part (TeleLab Frontend) that consists of web based application that allows navigation through learning units, deliver content and keeps track of the participant's progress. This TeleLab Frontend has access to database for user authentication, learning content, etc. The TeleLab Frontend consists of two parts. The first one (Admin Frontend) supports administrators who provide content, prepare the training environment, and monitor the system in general. The other one (User Frontend) is used by the Trainees to work through the learning units and to access the virtualized training environment. The TeleLab Frontend is connected to the back-end part (TeleLab Backend) to provide Virtual Laboratory for Internet Security hands-on exercises. The TeleLab Frontend is a Grails application, using an XML-RPC client to communicate with the TeleLab Backend.

The TeleLab Backend developed in ruby and implements an XML-RPC server for platform independent access. It receives requests from the TeleLab Frontend to create a virtual training environment for a Trainee (student). The virtual training environment consists of containers and a virtual switch. This virtual training environment known as a Team. Virtual switch is needed to create VLAN for isolating this virtual training network from other networks. Each container is provided with several tools which are needed to do the training. Docker is used as a platform to run container. The TeleLab Backend creates the virtual training environment by executing shell commands on the host where Docker Client installed and running. In docker system, shell command needs to be executed on a Docker Client. Commands for creating virtual training environment is listed in a script (ruby and shell). Which scripts to run is depending on the learning unit. This information is provided in the database. A detailed view of TeleLab Backend implementation is given in Section III-A.

A container start-up time is far quicker compared to a VM. The Container can be started up whenever a trainee submit a request for it, and it will be shut down whenever the trainee leaves the virtual training environment. The container life cycle will be described in section III-B.

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands you would normally execute manually in order to build a Docker image. By calling docker build from your terminal, you can have Docker build your image step by step, executing the instructions successively [13].

On Docker client, there are some Dockerfiles which are needed to generate container images. Image is needed to start a container. Dockerfile is created by the Trainer or Admin for each container image, based on the container roles in the virtual training environment. Trainer or Admin builds container image by executing a shell command on Docker Client, via TeleLab Backend.

A Container can be run with specific capabilities or even as a privileged container. Capabilities of container can be specified as needed. [14] explains the docker's capabilities. Docker Daemon runs and manages a container. Each container is separated by using namespaces, cgroups and UFS [14]. Docker container
consists of binaries and libraries that can be used to run an application. The Docker client is the primary user interface to Docker. It accepts commands from the user and communicates back and forth with a Docker daemon [14].

To generate a container, Docker Client runs a shell script on a Docker Daemon in a host machine. The host machine could be the main VM or another VM in the same cloud. When the needed containers are ready to be used, the Tele-Lab will open a GateOne page which provides an SSH connection to the attacker machine in the exercise scenario. To be able to access the virtual host in the public cloud, every attacker container is provided with an SSH server. A participant could access the virtual host using GateOne and SSH connection to the container. GateOne is a text-based remote access. If a desktop based host is used, then the Tele-lab will show a NoVNC connection to the participant. In the context of MITM attack and Firewall learning unit, text-based remote access is used.

A. TeleLab Backend

In Fig. 1 TeleLab Backend receives request from TeleLab Frontend to provide a virtual training environment for a student. To process the request, it needs to find out about which docker images to be used, the number of instances, IP address allocation, bridge number, VLAN ID and script to run. These information are stored in the database. Using information from the database, it sends commands to docker to create container instances. To start a container instance, a command needs to be executed on the host where a docker client exists. To generate a complete training environment for a participant, several commands need to be executed. These commands are listed in a shell script. The Tele-Lab Backend calls and executes this shell script. Each training environment has its own shell script.

VLAN is used to isolate each training environment. Every participant has one training environment where one environment has at least 2 containers in the same VLAN. With VLAN, each environment is a broadcast domain. This is enough to isolate un-necessary traffic from entering another training environment. For example, in the Ettercap training environment, with VLAN, the MITM attack on one training environment will not have an impact on another training environment. The Docker default switch was not designed to provide a VLAN, we need to use OpenvSwitch (OvS) to isolate a training environment using VLAN. [15] describes how to use OvS and VLAN configuration to isolate a training environment.

Because of VLAN implementation, IP addresses cannot be assigned using a DHCP server. Each VLAN is an IP subnet. IP address allocation for each VLAN are stored in the database. Before executing the script to create a training environment, Telelab backend system has to look for a range of IP addresses that are available to be used by the containers.

To run a docker container, a shell command needs to be executed. To create a training environment (Team) some docker commands in a form of shell script needs to be executed. Telelab backend receives a learning unit scenario ID from Telelab frontend. This Learning unit scenario ID is used to generate a training environment. The scenario ID has a map to Dockerfiles, container images, shell script. The Data model of Tele-Lab backend is shown in Fig. 2. A shell script is needed to create a Team or a training environment. This shell script starts container instances and assign Network (VLAN, bridge and IP addresses) to the container instances. After the Team is created, the user ID is mapped to a Team, VLAN, IP Address, container instances and Bridge ID. When the user stops the Team, container instances will be shut down and VLAN and IP Addresses will be released.

![Figure 2. Data model of telelab backend.](image)

B. Training Environment (Team) Life Cycle

A training environment is created when a user request for it, and it will be shut down when the user leaves or ends the remote access session. Fig. 3 shows the message flow in the process of creating a training environment. A user click a link on a browser to start a Tele-Lab hands-on exercise for a Learning Unit. The request is accepted by the Tele-Lab Front End. Tele-Lab Front End will verify the request and if the request is Ok, the request will be sent to the Tele-Lab Backend. Tele-Lab Backend check on the database, whether the images for the requested learning unit has been made or not. If the images is not available, the Tele-Lab backend will choose and run the specific Dockerfile to create images for this specific learning unit. After the images are ready, Tele-Lab backend will execute a script on the Docker Client to create a container with specific capabilities. Configuration setting such as capabilities, Dockerfile, for every learning unit is stored in the database. Commands or script in Docker Client is sent to Docker Daemon to be executed. Docker Daemon could be in the same VM with Docker Client or in another VM in another Public/Private Cloud. Docker Daemon will create containers based on the script. For example, in the MITM Attack exercise scenario, Docker Daemon creates two containers using script. One container has a role as a victim, which periodically sends credential to an FTP server. The other container is the attacker container where a participant has access to it and from this container, an attacker can run an attack using tools that has been provided in it. The attacker will try to get the victim's credential and use the credential to access the server and get specific file from the server to prove that he has successfully made the attack.

After the container was created, run and ready to be used, the Tele-Lab will provide remote access to the
participant. Remote access could be using noVNC for remote desktop or using GateOne for text based remote access. In case of text-based remote access, Tele-Lab will give a link to the user, where by clicking this link, the system will direct the user to the GateOne with ssh connection to the attacker container where the user can start an exercise. An Example of the link to GateOne SSH connection is https://192.168.56.205/?ssh=ssh://johannes@localhost.

### IV. SECURING THE PRIVILEGED CONTAINER

The Container is less secure than VM [16]. Access capability for a container must be given only to be able to run the application in it (principle of least privilege). Every module must be able to access only the information and resources that are necessary for its legitimate purpose. Access level can be given by using capability or privileged in run command options. Some applications need root access. For these applications that need root access, privileged access should be given.

Placing a training environment with privileged container in a public cloud must be designed carefully to prevent the container user from endangering the host. In a public cloud, the customer must obey the TOS (Term of Service). If it is really needed to provide privileged container, the system must be designed to prevent the container user from harming the host or other container. And if the system cannot prevent the privileged container from harming the host, we should also combine private and public cloud to provide a privileged container, where privileged container will be created in a private cloud and another un-privileged container can be created in a public cloud.

In MITM Attack training environment, container must be run with privileged. Running container with privileged is the same as running an application with root account in an operating system. A vulnerability in the application can lead to a malicious user getting access into the system. To do MITM attack, Ettercap is used as an attack tool. We assume that Ettercap and container has no vulnerabilities that can be used to get access to the host. But still precaution steps should be taken to reduce the risk. The container user should not be able to (1) access Internet (to prevent it from installing another application or download any kind of tools), (2) compile a source code (any compiler must be removed), (3) use any other application besides the designated one (unused applications must be removed).

### V. MITM ATTACK AND FIREWALL

The Tele-Lab has learning units where each one of them uses certain tools to do the exercises. Therefore, each learning unit needs a different training environment. To replace VMs with containers, we need to check whether each learning unit tool can be implemented in a container instead of a VM. In this paper, we only show 2 learning units: MITM attack and Firewall.

![Figure 3. Training environment life cycle.](image)

The allocated container instances will be stopped when the user no longer needs it (when the user logs out, or exit from GateOne). When the user logs out from the container instances, the system will stop all the instances related to the user exercise. This way the allocated memory and CPU for these container instances, will be released and ready to be used by others.

MITM attack is delivered by doing active eavesdropping where the attacker able to intercept and relay messages between victims. The attacker pretends to be the communicating partner of the victims, making the victims believe that they are talking directly to each other [17]. MITM attack has several attack model to intercept traffic communication between two communication partners. In this scenario, we used ARP (Address Resolution Protocol) Spoofing to do the attack. Fig. 4 shown MITM attack in action. Alice is communicating with Bob where Bob is an FTP Server. It is designed that periodically, Alice will send credentials to Bob. The attacker does MITM Attack to catch the credential. The credential will be used by the attacker to get a specific file from the FTP server. By doing ARP spoofing, an attacker can pretend to be Bob for Alice and vice versa. All traffic between Alice and Bob is being relayed via the attacker computer. While relaying, the messages can be captured and/or manipulated.
ARP is a protocol to resolve IP Addresses to MAC Addresses in a local area network (LAN). When Alice's computer opens an IP-based connection to Bob's computer in the local network, it has to determine Bob's MAC Address at first, since all messages in the LAN are transmitted via the Ethernet protocol (which only knows about the MAC Addresses). Alice broadcasts an ARP request to the local network and asks, "who has the IP Address of Bob?" Bob's computer answers with an ARP reply that contains its IP Address and the corresponding MAC Address. Alice stores that address mapping in her ARP cache for further communication.

ARP Spoofing is about sending forged ARP replies. The attacker sends ARP replies to Alice with Bob's IP Address and attacker's MAC Address. Refer to TCP/IP stack, at the Internet layer Alice has the true Bob's IP Address, but at the network access layer, Alice has the MAC Address of the attacker. Alice believes that the attacker's MAC address is Bob's MAC address. So, when the packet is sent to the network, the packet will be sent to the attacker. The attacker takes the packet and forward it to Bob. Vice versa, the same thing happened to Bob, where Bob has the MAC address of an attacker as the MAC address of Alice. To be able to communicate with Alice and Bob, the attacker has their true MAC Addresses.

IPTables (Firewall). IPTables is the default firewall tool available in Linux. Tele-Lab learning unit uses IPTables as the tool to secure a host or as a firewall to secure a network. In this training environment, we only need two containers. One of the containers is acting as a host to be protected. The other is as the host that generates traffic to the protected host. IPTables is installed in the protected host to allow only specific traffic that could reach the host. These two containers must be able to be accessed by the participant. Participant needs to configure the IPTables in the protected host and need to test the IPTables configuration from the other container. IPTables should only allow ssh connection to the protected host, other incoming traffic must be blocked.

VI. EXPERIMENT

The objectives of our experiments are (1) to prove that the Docker container can actually be used as a virtual host in Tele-Lab, (2) to find out whether it can increase the scalability or not. We use MITM attack and Firewall learning unit in our experiment, to represent other learning units in Tele-Lab. In our experiment we did not test it using real life user access. We generated the traffic in a local area network (LAN). When Alice's computer opens an IP-based connection to Bob's computer in the local network, it has to determine Bob's MAC Address at first, since all messages in the LAN are transmitted via the Ethernet protocol (which only knows about the MAC Addresses). Alice broadcasts an ARP request to the local network and asks, "who has the IP Address of Bob?" Bob's computer answers with an ARP reply that contains its IP Address and the corresponding MAC Address. Alice stores that address mapping in her ARP cache for further communication.

For the objective number one, we installed and configured Ettercap for the MITM attack learning unit and IPTables for a firewall learning unit on a docker container. These tools can be run well in the docker container. For the objective number two, we run the training environment (Team) as many as possible on the certain host to find out how many students can simultaneously do the exercise. We create a script and use cron job to simulate real user activities in doing the exercise.

We delivered our experiment on a VM. This VM was running in a Virtual Box hypervisor with resource allocation of 1.5 GBytes memory, 25 GBytes storage and 1 CPU. We use htop to monitor the resource consumption. As explained in V, there are 3 nodes are involved in MITM attack exercise scenario. In our experiment we use only one FTP server (Bob) for every training environment. For Alice container, we create a Dockerfile (Fig. 5) to create docker image that has an FTP client inside. Using crontab script (Fig. 6), Alice periodically (every 5 minutes) sends credentials to Bob. For the attacker container, we create docker image (Fig. 7) that has an Ettercap and SSH server installed and configured. The Ettercap will be executed using cron job (Fig. 8). The Ettercap command is stored in a shell script where we add a sleep command to randomly execute the shell script.

To create a MITM attack training environment for a student, only Alice and attacker containers that need to be created. We started the experiment by creating 20 training environments for 20 students, and gradually increased by 20 if the performance was still good. After the environments are created, we manually added an environment and tried the exercise. The performance is still good if there was no error shown up from the system, and we could do the exercise comfortably.

Until 100 training environment for 100 students, the performance was still good. But for 120 training environments the performance are not good anymore. The Linux system showed several errors of Killing processes because of “out of memory”. Htop showed that the CPU and memory were overloaded. In this condition,
we cannot create a new training environment. From this result, we concluded that for the VM of 1.5 GBytes memory, 100 students can be served to do MITM attack exercise. If we use VMs instead of containers, the available resource will only be enough for 1 student.

For Firewall learning unit, we only need two containers for a training environment. For the firewall container, we create a docker image that has IPTables, an SSH server, and an FTP server. IPTables has already configured to allow only SSH connection. For the other container, we create a docker image that has an ssh client, an FTP client and a Telnet client. These clients are used to test the IPTables configuration. We start the experiment in the same way as in the MITM attack. Until 200 training environments, the performance was still good. We stop the experiment, because we thought it was enough for the experiment objectives.

From these experiments, we can see that the memory requirement to run an application in a training environment plays the main role in the number of training environment that can be run on a host. Ettercap needs much more memory compare to IPTables. That is why, we could run more than 200 IPTables training environments, while we could only run 100 Ettercap training environments on the same host. To design and allocate resources for all Learning Units in Tele-lab, we need to find out the number of training units that can be run for each Learning unit exercise in the same host.

In these experiments we used a normal Linux OS without changes. In the future, we should use tiny Linux, to be able to serve more student, because tiny Linux uses less memory.

In the exercise using Ettercap, the CPU is always used 100%, even when there were only 20 training environments. This condition did not generate error and the training environments are still working well. The 25 GByte storage capacity was also not an issue in the experiments.

The same host can only be used for one training environment (one student) if we used VM as a virtual host in the training environment. So, from these experimental results, we can see that containers can be used to replace VMs and can significantly increase the scalability.

VII. CONCLUSION AND FUTURE WORK

The work described in the paper at hand shows that container can be used to replace VM as a host in Tele-Lab. It is also shown that the scalability can be increased. The container-based Tele-Lab system has been successfully implemented.

For the future work on this topic will be the extensive evaluation of the container-based Tele-Lab system with the real life user access. Another future work would be related to the security of the system, because we have to use privileged container. We need to analyze deeply the risk of using a privileged container for a virtual host, and find the best way to secure it.

There is a possibility that the host can be tampered by the privileged container. Providing agent on the host can detect when the host is being tampered by the privileged container. The agent must monitor every access or any strange activities from the containers to the host. This agent will inform the sysadmin whenever it detected an attempt of attack. It can also be used as a measurement of the security of the cloud provider. The implementation of this idea will be described in our next paper.

REFERENCES


Johannes Harunguang Sianipar is a Ph.D student at Hasso Plattner Institut, University of Potsdam Germany. His area of research is virtual laboratory for Internet security e-learning. He is also interested in cloud trust. He studied computer science at Institut Teknologi Bandung Indonesia. In 2000, he received his master degree with the title M.T. He was a lecturer at Institut Teknologi Del
Christian Willems studied computer science at the University of Trier. He graduates in 2006 as a computer scientist (thesis: "A generic interface for virtual machine monitor as an extension of Tele-Lab infrastructure"). He is a research associate at the Hasso Plattner Institute. He teaches in the areas of Internet technologies, security and virtualization. In addition, he is working on his doctoral thesis at the chair of Prof. Dr. Christoph Meinel. His research interests focus on Awareness Creation, teaching in the IT security and virtual machines and cloud computing. Since 2012, he has been as technical director responsible for the operation and development of the MOOC platform of the Hasso Plattner Institute - openhpi, the Teaching Team advises for openhpi courses in HPI and coordinates cooperation with openSAP.

Professor Dr. Christoph Meinel (Univ. Prof., Dr. sc. nat., Dr. rer. nat., 1954) is Scientific Director and CEO of the Hasso Plattner Institute for Software Systems Engineering GmbH (HPI). He is a member of acatech, the German “National Academy of Science and Engineering,” and numerous scientific committees and supervisory boards. He is a full professor (C4) for computer science and serves as department chair of Internet Technologies and Systems at HPI. His areas of research focus on Internet and Information Security, Web 3.0, Semantic Web, Social and Service Web and the domains of e-learning, teleteaching and telemedicine. He is scientifically active in innovation research on all aspects of the Stanford innovation method “Design Thinking.” His earlier research work concentrated on the theoretical foundation of computer science in the areas of complexity theory and efficient OBDD-based algorithms and data structures. Christoph Meinel is author/co-author of 9 books and 4 anthologies, as well as editor of various conference proceedings. More than 400 of his papers have been published in high-profile scientific journals and at international conferences. He is also editor in chief of “ECCC – Electronic Colloquium on Computational Complexity,” “ECDTR – Electronic Colloquium on Design Thinking Research,” the “IT-Gipfelblog” and the tele-TASK lecture archive and openHPI.