Proposal No. 830929
Call H2020-SU-ICT-03-2018

Project start: 1 February 2019
Project duration: 42 months

Cyber Security for Europe

D7.2

Virtual lab for open-source tools education and research

Document Identification	
Due date	31 December 2020
Submission date	30 December 2020
Revision	0.2

Related WP	WP7	Dissemination Level	Public
Lead Participant	BRNO	Lead Authors	Jan Vykopal (BRNO) Valdemar Švábenský (BRNO) Attila Farkas (BRNO)
Contributing Beneficiaries	---	Related Deliverables	---

Abstract:

The deliverable D7.2 consists of two artifacts: this report and the implementation of the developed tool stored in a repository at https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator. The report presents the requirements for Cyber Sandbox Creator, a tool for creating open-source, portable, and lightweight virtual labs for cybersecurity education, testing, and certification. These labs are aimed at individuals and small and medium organizations. The text describes the existing technologies that can be used as building blocks for lab development. It also details the architecture of the lab that we create to serve educators, researchers, and other stakeholders in the cybersecurity domain. Cyber Sandbox Creator incorporates Vagrant, Ansible, and VirtualBox as the technologies proven in practice. We also outline three demonstration use cases to validate and test the proposed lab.

This document is issued within the CyberSec4Europe project. This project has received funding from the European Union's Horizon 2020 Programme under grant agreement no. 830929.

Executive Summary

This report introduces an open-source tool for building lightweight virtual laboratories for cybersecurity education, testing, and certification. First, we list the relevant user roles and scenarios in which these users might need such a tool. One of them is an educator, who can efficiently prepare training content for a group of students, or an auditor, who can quickly set up a testing environment even with limited knowledge about virtual machines.

Then, we review similar existing tools relevant for the scenarios, such as other lightweight, locally virtualized labs and cyber ranges. We argue that these tools fulfil the needs of the mentioned user roles only partially, but not entirely. Therefore, we present building blocks for our proposed tool, Cyber Sandbox Creator (CSC), and discuss their advantages and disadvantages. After that, we introduce the architecture of CSC and the virtual lab, including definitions, underlying technologies, and example usage. Finally, we show three use cases of applying CSC in developing a testing environment, educational activities, and certification activities.

CSC is released as open-source software available at https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator.

Document information

Contributors

Name	Partner
Vashek Matyáš	BRNO

Reviewers

Name	Partner
Jozef Vyskoč	VaF
Bruno Crispo	University of Trento

History

Version	Date	Authors	Comment
0.1	2020-09-10	Jan Vykopal Valdemar Švábenský Attila Farkas	Enhancement of the D7.2 interim report "Architecture of a virtual lab" v1.0 sent to internal reviews. The first release of Cyber Sandbox Creator software, available at https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator/-/releases/v1.0.0.
0.2	2020-12-18	Jan Vykopal Valdemar Švábenský Attila Farkas	Incorporated comments from reviewers. The minor update of Cyber Sandbox Creator software based on the reviews. Available at https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator/-/releases/v1.0.1
0.2	2020-12-30	Ahad Niknia	Final check and prepareation for submission

Table of Contents

List of Figures

List of Acronyms

B	**BRNO**	Masaryk University, Brno, Czech Republic
C	**CIDR**	Classless Inter-Domain Routing
	CS4E	Cyber Security for Europe
	CSC	Cyber Sandbox Creator
	CTF	Capture the Flag
E	**ELK**	Elasticsearch, Logstash, Kibana
	EU	European Union
H	**HW**	Hardware
I	**ICT**	Information and Communications Technology
	IP	Internet Protocol
	IT	Information Technology
J	**JAMK**	Jyväskylän ammattikorkeakoulu (University of Applied Science), Jyväskylä, Finland
K	**KVM**	Kernel-based Virtual Machine
L	**LTS**	Long Term Support
M	**MS**	Microsoft
O	**OS**	Operating system
P	**PC**	Personal Computer
R	**RAM**	Random-access Memory
	RGCE	Realistic Global Cyber Environment
S	**SSH**	Secure Shell
	SW	Software
U	**USB**	Universal Serial Bus
V	**VM**	Virtual machine
Y	**YAML**	YAML Ain't Markup Language

Glossary of Terms

C **Capture the Flag game, CTF game**

A training activity in which participants solve technical cybersecurity tasks. Completing each task results in finding ("capturing") a text string called flag.

Cyber range

A cyber range is a platform for the development, delivery and use of interactive simulation environments. A simulation environment is a representation of an organisation's ICT, OT, mobile and physical systems, applications and infrastructures, including the simulation of attacks, users and their activities and of any other Internet, public or third-party services which the simulated environment may depend on. A cyber range includes a combination of core technologies for the realisation and use of the simulation environment and of additional components, which are, in turn, desirable or required for achieving specific cyber range use cases. [ECSO]

S **Sandbox**

An isolated virtual environment that allows cybersecurity experimentation (including the execution of cyber attacks) without negative consequences on the underlying IT infrastructure.

Cyber Sandbox Creator

A tool developed as a part of the CS4E project that allows the user to create arbitrarily defined sandboxes.

Sandbox Runner

A tool developed as a part of the CS4E project that allows the user to run the sandboxes created with the Cyber Sandbox Creator.

T **Training**

A series of hands-on cybersecurity tasks completed in a virtual environment under the guidance of a human or automated tutor. The learner's goal is to develop and practice cybersecurity skills. The training can have various formats, most commonly a cybersecurity game, in particular Capture the Flag game.

V **Virtual lab**

The deliverable of the CS4E project that includes Cyber Sandbox Creator and Sandbox Runner.

Virtualization hypervisor

Software and hardware that creates and runs virtual machines.

Virtualization orchestrator

A tool for coordination and management of multiple virtual machines.

Virtualization provisioner

A tool for automated configuration and installation of software into virtual machines

1 Introduction

Digital skills are recognized as a highly-valued competence of current and future workers, regardless of their work role or employment sector. European Commission considers them the cornerstone of a truly functioning digital society, Digital Single Market, and a key for boosting the EU's competitiveness.

A 2016 survey indicated that concerning ICT professionals, there could be 8.7 million ICT professionals in the EU and 756,000 unfilled jobs in 2020. Almost 240,000 IT graduates keep entering the labour market each year, and more than 100,000 new IT practitioners enter without a formal degree [EUROPA19].

A more recent (ISC)² Cybersecurity Workforce Study [ISC2] estimates 291,000 unfilled cybersecurity jobs in Europe and about 4 million globally in 2019. In Europe, the gap has almost doubled since 2018. Increasing hiring demand was seen in smaller companies with 1 to 99 employees and companies with 500+ employees [ISC2].

All these figures indicate two needs in the cybersecurity domain. First, there is a need for training future cybersecurity experts and professional workers who enter the cybersecurity field from other domains. Cybersecurity workers must be equipped with operationally-relevant skills that can be gained and exercised most efficiently in a hands-on manner. Second, since cybersecurity involves the creation, operation, analysis, and testing of secure computer systems [JTF], many cybersecurity work roles require using specific tools to efficiently fulfil their day-to-day tasks. For instance, a malware analyst needs a sandboxed virtual environment. In addition, the emerging EU framework for the ICT certification of products and services introduced in the EU Cybersecurity Act [ENISA] will strengthen the need for a suitable lab environment.

All these needs can be addressed by using cyber ranges. These are interactive, simulated representations of an organization's local network, system, tools, and applications that are connected to a simulated Internet-level environment. They provide a safe and legal environment to gain hands-on cyber skills and a secure environment for product development and security posture testing [NIST]. However, some use cases do not require all their features, and using a cyber range is time and cost-prohibitive, particularly by individuals or in small and medium organizations.

To cater to users who cannot afford operating complex and costly full-fledged cyber ranges, we propose a lightweight virtual lab environment for cybersecurity education, testing, and certification. Our design goal is to be able to run the lab environment at a single computer or laptop using state-of-the-art free and open-source components and best practices. This approach will enable more users and companies to participate in cybersecurity education, research, and development using their existing computing resources with no additional costs.

1.1 Document Outline

This document is structured into seven sections. Section 2 presents user requirements, user roles, and stories for using the virtual lab environment. Section 3 refers to related work in lightweight virtualized labs, cyber ranges, and Capture the Flag platforms. Section 4 introduces available building blocks for the virtual lab. Only the building blocks relevant for the defined requirements are presented. Section 5 describes the architecture of the proposed lab environment. Section 6 outlines demonstration cases of the proposed lab. Finally, Section 7 summarizes current progress and work plan.

2 User Requirements

We now define several user roles for those who want to use a virtual lab environment for their cybersecurity-related missions. Section 2.1 presents these roles, along with their general constraints and non-functional requirements for the virtual lab in the form of user stories. This approach enables us to focus on shared requirements for the lightweight, virtual lab which is delivered with this document. These requirements are then defined in Section 2.2.

2.1 User Roles and Stories

We identified six user roles: Educator, Trainee, Researcher, Developer, Specialist, and Auditor. All roles share the constraints that they have limited time and budget. Therefore, they all need a user-friendly interface of the lab, quality documentation, and low entry requirements for usage. The specifics of the roles are explained below, together with relevant user stories. The stories are traditionally written in the form *As a [type of user], I want [an action] so that [a benefit/a value]*.

2.1.1 Educator

An Educator is a teacher (instructor), regardless of the level of the education system. This role includes teachers in educational institutions, as well as in professional education and extracurricular events (competitions, summer camps). The Educator has the following constraints and needs:

- Educator needs ready-made training components, such as tutorials, exercises, games, or VMs. An Educator will use these building blocks to create lessons suitable for his needs.
- Technical details of the inner workings of the virtual environment are not relevant for the Educator. He may even lack the technical knowledge to understand them.
- Educator is unlikely to customize the lab. He needs reasonable defaults.
- Educator may want to receive classroom orchestration reports from an ongoing training session.
- Educator may want to use the lab for the assessment of students' learning.

Educator's user stories:

- As an Educator, I want to automatically instantiate existing training content for (a group of) Trainees so that I save my precious time.
- As an Educator, I want to see the progress of Trainees in an ongoing session so that I can better facilitate the training session.
- As an Educator, I want a report from a finished training session so that I can use it for formative or summative assessment of Trainees.

2.1.2 Trainee

A Trainee is any person involved in educational activities that are prepared and/or supervised by the Educator. The role encapsulates students, professional learners, and cybersecurity enthusiasts who wish to enhance their skills.

The Trainee has the following constraints and needs:

- Trainee wants to learn as much as possible in little time.
- Trainee wants to learn knowledge and skills relevant to his environment (e.g., university curriculum, workplace needs).
- Trainee wants to learn in an engaging way.

- Trainee may require formative assessment (feedback), especially during practice exercises.
- Trainee may require summative assessment (grading), especially during competitions or workplace skill evaluations.
- Trainees may work in groups to practice teamwork and soft-skills.

Trainee's user stories:

- As a Trainee, I want to see mistakes I have made and an overview of my progress at the end of training so that I can see what I did well and what I should improve.
- As a Trainee in a competition, I want to see the comparison of my skill level to the skill level of others so that I can see how well I performed.

2.1.3 Researcher

A Researcher is any person involved in cybersecurity research projects, whether it is for academia, industry, or government. He specializes either in research of cybersecurity itself and/or in cybersecurity education. The first one can be concerned, for example, with the behavior of the infrastructure (e.g., attack detection in the network). The second one focuses particularly on educational data mining and learning analytics of data from cybersecurity training.

- Researcher poses a wide range of cybersecurity skills.
- Researcher may have a (small) dedicated budget for the research infrastructure or access to in-house (cloud) infrastructure.
- Researcher needs a lab with customizable building blocks to adjust it for his research.
- Researcher needs a lab environment that is isolated from the outside environment.
- Researcher needs the feature of logging the interactions with the lab environment, its internal processes and the support for processing and analyzing the logs.

Researcher's user stories:

- As a Researcher, I want to quickly set up a custom virtual infrastructure (which may include connecting external devices and custom SW) so that it fits my specific needs.
- As a Researcher, I want to quickly set up a virtual infrastructure based on ready-made components and existing templates so that I can focus on my actual research.
- As a Cybersecurity Researcher, I want to collect data about the infrastructure itself so that I can analyze its properties and behavior.
- As a Cybersecurity Education Researcher, I want to collect data about trainees' progress and interaction with the training environment so that I can analyze them.
- As a Researcher, I want to export the data in a standardized format so that I can use it in further analysis outside the virtual lab.

2.1.4 Developer

A Developer is a person creating a cybersecurity product (hardware or software).

- Developer may not always have extensive cybersecurity skills, only software development skills.
- Developer needs a lab that can be used for regression tests of the developed software.
- Developer has access to in-house (cloud) infrastructure that can host the lab.

Developer's user stories:

- As a Developer, I want to quickly set up a custom virtual infrastructure (which may include connecting external devices and custom SW) so that I can test my product.
- As a Developer, I want to quickly set up a virtual infrastructure based on ready-made components and existing templates so that I can focus on actual testing.
- As a Developer, I want to easily deploy a new build of my product in the virtual infrastructure so that I can test fixed bugs or new features.

2.1.5 Specialist

A Specialist is a cybersecurity professional, such as an analyst, penetration tester, or incident handler.

- Specialist possesses a cyber operations skill set, which includes especially Data Security, Network Security, and System Security knowledge areas from the JTF Cybersecurity Curriculum [JTF].
- Specialist needs a lab for post-mortem, quick-and-dirty analysis in day-to-day operations.
- Specialist has access to in-house (cloud) infrastructure.

Specialist's user stories:

- As a Specialist, I want to quickly set up a custom virtual infrastructure (which may include connecting external devices and custom SW) so that it fits my current needs.
- As a Specialist, I want to quickly set up a virtual infrastructure based on ready-made components and existing templates so that I can focus on actual work.
- As a Specialist (analyst), I want to easily deploy an unknown artifact in the virtual infrastructure so that I can analyze it. The unknown artifact can be software, files, or a hardware device connected via peripheral ports such as USB.
- As a Specialist (pentester), I want to launch attacks in an isolated infrastructure so that I can examine their properties.

2.1.6 Auditor

An Auditor is a member of a certification lab that verifies the security properties of a software or hardware product.

- Auditor has a (limited) cybersecurity skillset.
- Auditor needs a lab that can be used for certification of a wide spectrum of systems.
- Auditor has access to in-house (cloud) infrastructure.

Auditor's user stories:

- As an Auditor, I want to quickly set up a predefined virtual infrastructure for repetitive testing of external HW devices and custom SW.
- As an Auditor, I want to design and run test suites against external HW devices and custom SW.
- As an Auditor, I want to get test reports from testing external HW devices and custom SW.

2.2 Requirements for the Proposed Lab

After careful consideration and analysis of various user roles and stories described above, we derived the following requirements for the proposed virtual lab.

Properties of the lab itself:

1. Lab must be easy to use. Instantiation and disposal of the lab environment must be as automated as possible (ideally in one click). The lab must prevent its users from making common mistakes. [easy to use]
2. Lab environment is built with minimal overhead so that it can be run at a single commercial off-the-shelf computer. [minimal system resources required]
3. Lab environment is built using free and open-source components, no additional fee[1] is required for using any component at common operating system families (Windows, Linux, macOS). [free and open-source components]
4. Lab enables users to run hosts with operating systems different to the operating system of the host computer. [interoperability]
5. Lab supports running a distributed training session, i.e., each trainee working at their own host with the lab. [distributed training]
6. Lab enables access to hardware connected to the host at which the lab is running. [access to hardware]

Properties of the virtual environment instantiated using the lab:

7. Lab instantiates networks of full-fledged virtual machines, i.e., it emulates the networked hosts that run a standard operating system. [full OS emulation]
8. Lab allows controlling the level of isolation of the created environment from outside. The environment can be completely isolated as well as connected to the Internet. [isolation]
9. Lab can be suspended and its internal state saved. It can be later recovered to its previous state. [suspendability]
10. Lab supports logging of various events (such as commands typed by learners or researchers), either locally or at a remote central endpoint, so that users can track the ongoing and past processes in the lab environment. [logging]

Using the lab to define a virtual network:

11. The definition of the virtual network and machines must be described in structured formats that allow simple modifications according to various and dynamic user needs. [infrastructure as code]
12. The definition of the virtual network and machines must be well-documented and ready for sharing with others who would like to instantiate the same environment, either in the lab or in a full-fledged cyber range. [portability]

2.3 Limitations

Although these requirements do not cover all the user stories described in Section 2.1, we believe they enable the most common use cases at a considerably lower cost and overhead than full-fledged cyber ranges.

[1] The lab is composed from building blocks that are free of charge. However, if users deploy any commercial software within the lab, including an operating system, they are expected to cover the license fees.

Notably, the lab is intended for individuals dealing with several virtual machines in a small network rather than for large organizations running numerous virtual environments for different users at the same time. We focus by design on individual work in the lab environment deployed at a single computer. However, several Trainees can work with the lab by forming a group at one computer hosting the lab environment.

3 Related Work

The user requirements described in Section 2 are partially fulfilled by several already existing tools and systems. However, none satisfies all the requirements. The following sections summarize the most relevant available solutions.

3.1 Lightweight, Locally Virtualized Labs

A few other virtual lab environments already exist. They have similar functionality as the proposed lab, but they use different technologies and run only on a subset of platforms (for instance, Linux only). We detail their features and limitations below.

3.1.1 CyRIS

CyRIS (Cyber Range Instantiation System) [Cyris] is a project developed by the Japan Advanced Institute of Science and Technology (JAIST). It allows to instantiate cyber ranges from base OS images and a YAML configuration file, configure the created virtual machines, install arbitrary tools, and emulate incidents. The tool includes a shell script to automatically install the required dependencies to the host machine (or machines), including the virtualization hypervisor QEMU/KVM. SSH key management is not automatic. The user needs to generate a key pair before the configuration starts. The project recently added support for Windows host machines. Currently, base images for CentOS 7, Ubuntu 16.04 LTS, Ububtu 18.04 LTS and Windows 7 images are supported. Users can also create their base images (with specific requirements). The virtual network configuration is limited: only one network is created (automatically), and all hosts are part of that network. Logging of various events on each VM, which is one of our requirements, is also not supported.

3.1.2 Lability

Lability [Lability] is another similar solution for Windows, which uses Hyper-V to create virtual machines. It is written in PowerShell, and its input definitions are extended DSC documents (PowerShell Desired State Configuration). This tool can create virtual machines with specified virtual disk images and connect them with a predefined network topology. The tool can automatically download ISO images for Windows and install Windows updates or additional software. It requires Windows on the host machine to run, but besides Windows, some specific Linux distributions are also supported as virtual machines. This means that the requirement for interoperability is not fully satisfied. The project is not well-maintained, there were only few updates in 2020.

3.1.3 Labtainers

Labtainers [Irvine17] is a framework based on Docker containers that allows developing and deploying cybersecurity lab exercises. Similarly to the proposed lab, it allows emulating network environments without the need for extensive infrastructure. However, it does not support full virtualization, which is one of our requirements. The isolation is also weaker by using Docker containers. This framework was developed by the Naval Postgraduate School in Monterey, California.

3.2 Cyber Ranges

A cyber range is a simulated network environment that allows exercising cybersecurity skills and performing security testing. It extensively features virtualization, employs cloud computing, and often incorporates the building blocks mentioned in Section 4. Unclassified cyber ranges and security testbeds, such as KYPO or CRATE, are thoroughly discussed in a recent systematic literature review by Yamin et al. [Yamin20]. While this review describes only cyber ranges that were published in an academic paper, there are also commercial cyber ranges such as RGCE (by a CS4E partner JAMK), RHEA [RHEA], and Guardtime [Guardtime].

The main difference compared to lightweight, locally virtualized labs is that cyber ranges support running multiple and/or large-scale virtual environments with tens of networked hosts for many users at the same time. In contrast, lightweight labs focus on individual users working with a simple environment of several hosts. Besides, a cyber range often requires non-trivial underlying infrastructure, such as private cloud or bare-metal installations, and dedicated human resources for its operation and support.

3.3 Capture the Flag (CTF) Platforms

The virtual environment may lack the interface for educators and trainees that will guide and support these user roles through the training. This interface is usually provided by a CTF platform. Taylor et al. [Taylor17] present a comprehensive overview of 36 CTF platforms and their categorization based on content and availability. Similarly, Kucek and Leitner [Kucek19] compare the functionality of 28 CTF platforms, among which 12 are open-source, and 8 of these are examined closely.

The code of many CTF platforms is available on GitHub. One of those platforms that also contains an associated publication is PicoCTF [Chapman14]. Another popular framework used by many existing CTF games is CTFd [Chung17]. CTF-like challenges are so popular among cybersecurity enthusiasts that there are also commercial training platforms, such as Avatao [Avatao].

4 Existing Technologies as Building Blocks

In the previous section, we described multiple existing solutions. However, each of them has limitations that prevent them from fulfilling the user requirements defined in Section 2.2. Therefore, we will address these limitations by creating our own virtual lab. The lab will employ freely available, open-source, and well-tested components. In this section, we briefly introduce candidate technologies that can serve as building blocks for the proposed lab. Based on this summary we select the most suitable technologies for the proposed lab.

First of all, we need a virtualization platform for emulating the hosts and networks in the lab environment. The virtual machines will be managed by an orchestrator. The software running on the chosen operating systems will be configured and set up using a provisioner. We will also consider more lightweight alternatives to full-fledged virtualization, such as containers. The main components for building the virtual lab is depicted in Figure 1. When the machines, software, and data are ready for experimentation in the lab, they can generate logs that can be analyzed using a logging tool.

In addition to the lab environment, Educators and Trainees need a separate Capture the Flag portal in some use cases. The portal will serve task assignments and hints, whereas the lab allows working with the virtual hosts.

4.1 Virtualization Platforms

We need free software that is capable of running virtual machines from OS images. It should work on a regular PC running one of the major operating systems, so we require a type-2 hypervisor. It also must be able to run multiple virtual machines at once. [Techradar] includes a list of software for virtualization with a brief description. Here we describe the best candidates from this list, then mention the other, not suitable alternatives.

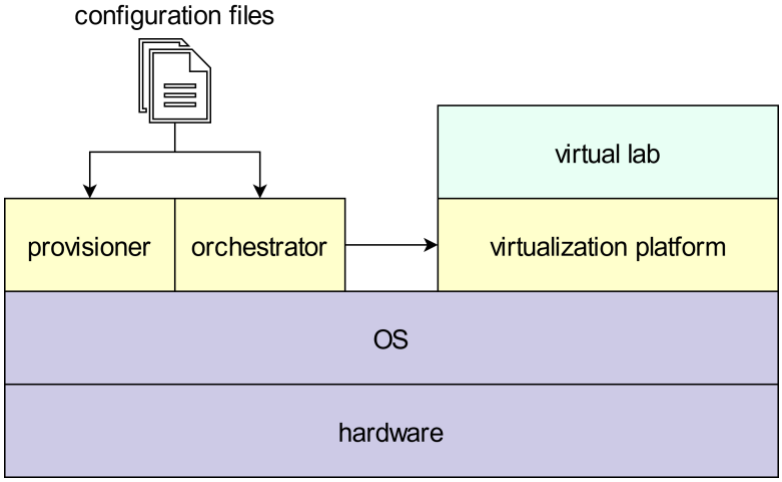

Figure 1: Main general components for creating and running a virtual lab

4.1.1 VirtualBox

VirtualBox [VirtualBox] by Oracle is an open-source hypervisor for virtualization. This product can create and manage guest virtual machines. It runs on every major operating system and can create Windows, Linux,

BSD, and other virtual machines. Users can load multiple guest OSs, and each guest can be configured independently. VirtualBox has a wide range of functions, including various types of virtual networking, support of hard disk images, and shared folders.

4.1.2 VMware Workstation Player

VMware Workstation Player [VMwarePlayer] is another virtualization hypervisor, which is a free version of the more feature-rich but commercial VMware Workstation. It supports Windows, Linux, NetWare, and Solaris virtual machines. This hypervisor is easy to use and fast but has fewer features than VirtualBox. Most importantly, the free version cannot run multiple virtual machines at once [Technologyadvice1].

4.1.3 QEMU/KVM

Kernel Virtual Machine [KVM] is a Linux kernel module that allows a program to utilize the hardware virtualization features of various processors. QEMU [QEMU], which is a generic and open-source machine emulator and virtualizer, uses KVM when running a target architecture that is the same as the architecture of the host. Unfortunately, it is for Linux only. Also, the combination of QEMU and KVM creates a type 1 virtualization tool [Cloudbuilder].

4.1.4 Other Free Tools

Techradar lists other popular projects that could be potential candidates [Techradar]. These projects are, however, not suitable for our purposes because of the following reasons. Microsoft Hyper-V has limited support for Linux. Xen Project is a type-1 hypervisor, which does not run on other OSs but rather directly on hardware.

4.2 Orchestrator

As the next component, we need a product that can automatically orchestrate the creation of all the virtual machines.

4.2.1 OpenStack

OpenStack [OpenStack] is a collection of open-source tools by RedHat for building and managing cloud computing platforms. It is a cloud operating system that controls large pools of computing, storage, and networking resources, all these managed through APIs with common authentication mechanisms. OpenStack requires at least two computers (a controller node and a compute node), each must have at least 8 GB of RAM and 100+ GB of space on the hard drive [OpenStackDocs]. It is possible, but a quite heavyweight alternative. Its users are, for example, CERN, T-Mobile, Adobe, American Airlines, and Walmart.

4.2.2 Vagrant

Vagrant [Vagrant] is a tool for automatized deployment of virtual machines. It is capable of creating virtual machines with given parameters, install operating systems on them using the provided images, and connect these machines to predefined virtual networks. This open-source tool supports a wide range of host operating systems (including Linux, Windows, and macOS). Vagrant can also work with multiple environments for virtual software development environments (VirtualBox, KVM, Hyper-V, Docker, VMware, and AWS). Its usage is simple: as an input, it takes one file (the Vagrantfile) and automatically creates every host that was defined in the input file. It can run on a single computer, and it is a more lightweight alternative than OpenStack because it adds almost no overhead to resources required by the created virtual machines. Companies like Mozilla, Nokia, or BBC use Vagrant.

4.2.3 Kubernetes

Kubernetes [Kubernetes] is an open-source system for automating deployment, scaling, and management of containerized applications. It coordinates a highly available cluster of computers that are connected to work as a single unit. Kubernetes automates the distribution and scheduling of application containers across a cluster in an efficient way. A Kubernetes node is a virtual machine or a physical computer that serves as a worker machine in a Kubernetes cluster. A cluster that handles production traffic should have a minimum of three nodes. Using more computers as one unit, Kubernetes does the opposite of what we are trying to do, which is creating a network of virtual machines on a single computer.

4.3 Provisioner

We also need a provisioner to configure the created virtual machines. It should also be free and open-source, and it should not require an agent on virtual machines. Not needing an agent makes it possible to use a broader range of OS images, not only the specialized ones with an installed agent.

4.3.1 Ansible

Ansible [Ansible] is an open-source tool for provisioning, configuration management, and application deployment. It can automatically configure virtual machines: install software, change files, or execute arbitrary commands. Together with Vagrant, they make an effective combination that can create and configure virtual machines using a single command.

Ansible works through SSH and requires only Python and OpenSSH to be present on the OS of the virtual machine. It runs on the host machine and configures the virtual machine over SSH, or it can also install itself on the virtual machine and run directly there. Input files (playbooks) use a descriptive language based on YAML and Jinja templates.

4.3.2 Chef

Chef [Chef] is a configuration management tool. It uses a domain-specific language for writing system configurations called *recipes*. Chef is used for maintaining servers in a company. It can integrate with cloud-based platforms such as Google Cloud Platform, Oracle Cloud, OpenStack, or Microsoft Azure and automatically provision and configure new machines but needs an agent on every virtual machine [Upguard2].

4.3.3 Puppet

Puppet [Puppet] is an open-source software configuration management tool. It can manage various stages of the IT infrastructure lifecycle, including provisioning, patching, configuration, and management of the operating system and application components across data centers. It runs on many Unix-like systems and Windows. For system configuration description, it uses its own declarative language. The configuration works only through agents installed on host machines [Upguard1].

4.4 Logging

After the core lab environment is functional, we need a tool for logging, which can gather data from multiple sources and visualize it later. A list of such tools with their properties is available at [Comparitech].

4.4.1 ELK

ELK [ELK] stands for Elasticsearch, Logstash, and Kibana. It is a combination of three open-source technologies: Logstash, a data processing pipeline, Elasticsearch, a search and analytics engine, and Kibana, a visualization tool. Logstash allows to pull data from one or more sources. Users can then search this data using Elasticsearch. Finally, the results can be graphically displayed in a Kibana dashboard.

As of August 2020, the projects are well-maintained: Logstach, Kibana, and Elasticsearch are very popular open-source projects (11.600, 14.700, and 50.500 stars on GitHub) with 453, 580, and 1.503 contributors, respectively [Logstash, Kibana, Elasticsearch]. It is trusted by companies like AirBus, Adobe, Cisco, eBay, Microsoft, Volkswagen, or Facebook [ElasticStories].

4.4.2 Fluentd

Fluentd [Fluentd] is a free and open-source tool that can collect live data streams to create log files as well as monitor and manage existing files. Results from log record analysis can be set to trigger alerts. However, these must be processed by Nagios, or a Nagios-based monitoring system, which would be an unnecessary restriction to the project. The core package can be extended through (usually free) plugins written by other community members. As of August 2020, the project on GitHub is well-maintained (9.5k stars and 200 contributors). Its customers are, for example, Microsoft, Amazon, or Nintendo [FluentdTestimonials].

4.4.3 Other Free Tools

There are more free tools on the list [Comparitech], but they have other limitations which make them insufficient. Papertrail, Splunk, and XpoLog pose limits on the amount of collected data per day (or month) on their free plan. ManageEngine Syslog Forwarder is Windows only, Graylog is Linux only, and Managelogs is not an active project anymore [Comparitech].

4.5 Capture the Flag Platforms

Among the CTF platforms examined in [Taylor17, Kucek19] (see Section 3.3), the open-source CTF platform picoCTF appears to be the best candidate. CTFd, another popular platform, also seems to be a good alternative. We will test these two in our future work.

4.5.1 picoCTF

PicoCTF [picoctf] platform is an infrastructure that is used to run the picoCTF game [picoctfgame], a security game targeted at middle and high school students, created by security experts at Carnegie Mellon University. The platform is designed to be easily adapted to other CTF or programming competitions.

4.5.2 CTFd

CTFd [Chung17] is a freely available and open-source platform for hosting Capture the Flag (CTF) challenges. It simplifies generic tasks that are necessary when creating any CTF event, such as specifying task assignments, hints, and solutions; forming teams; and displaying the score. CTFd runs as a standalone service that can be combined with any environment in which the player can solve the game tasks. It is used by many CTF games and a cornerstone for other CTF platforms like Yukti CTF [yuktictf], TJCTF [tjctf], and IceCTF [coldcorectf].

4.6 Containers

Containers allow deploying software on virtual machines more easily. A comparison of the best open-source solutions is available in [shadowsoft]. We mention only Docker here and leave a thorough comparison and testing of other candidates for our future work.

4.6.1 Docker

Docker [Docker] is a platform that allows to build, test, and deploy applications. It packages software into standardized units called *containers* that contain all the components that a software needs to run, including libraries, system tools, code, and runtime environment. These containers are isolated from each other and bundle their software and configuration files, but they can communicate with each other. A single kernel is used for all containers, so they are more lightweight than virtual machines.

5 Architecture of the Open-source Virtual Lab

Figure 2 depicts the underlying architecture of the proposed virtual lab. The lab environment is composed of several components running on a single host computer and optional components hosted remotely. The virtual lab can be run on standard operating systems with 64-bit x86 processor architecture, that is, MS Windows, Linux, and macOS. Detailed description of software requirements is present at the wiki page of the project https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator/-/wikis/home.

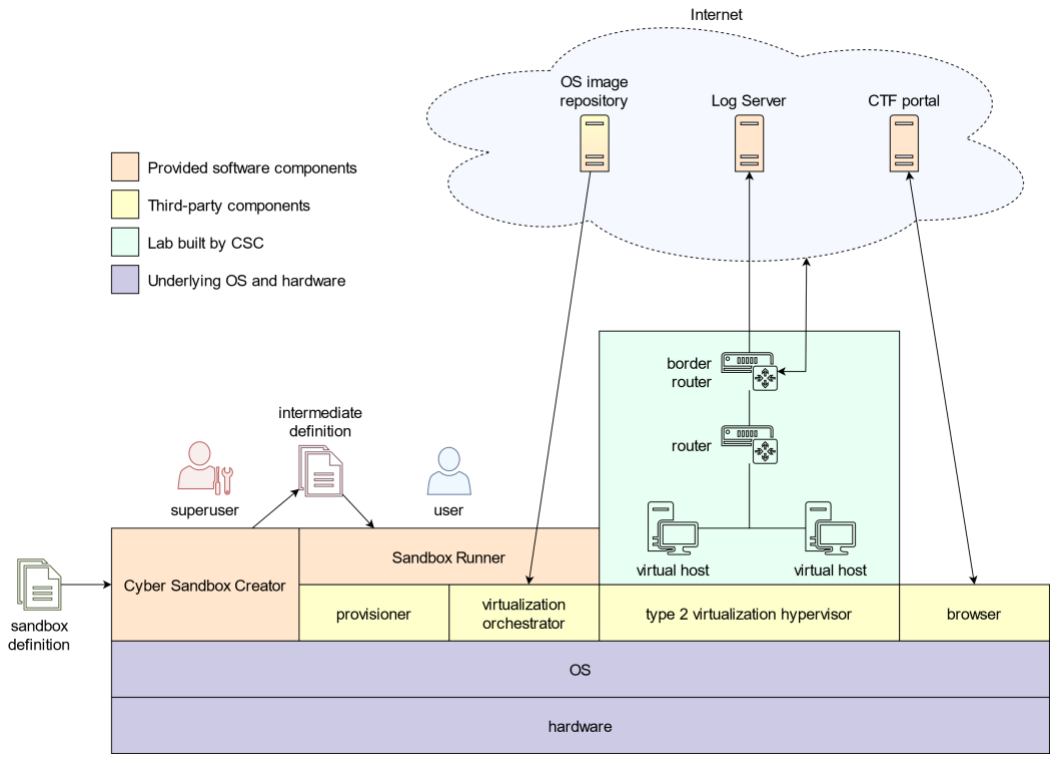

Figure 2: Architecture of the virtual lab, including the Cyber Sandbox Creator and essential infrastructure elements

The creation of a virtual lab environment starts with a *sandbox definition*. This definition covers everything from virtual machine parameters and configuration to network topology of the virtual lab. An advanced user[2] familiar with the syntax and semantics of definition files can create them to suit the needs of other users of that environment who be less experienced[3]. We further refer to this person as a *superuser*. A superuser takes these files and, using the proposed Cyber Sandbox Creator, generates the definition files that serve as input for the virtualization orchestrator and the provisioner. We will refer to these files as an *intermediate definition*. The superuser can edit these files, but more importantly, can distribute them to other

[2] That is an Educator, Researcher, Developer, Specialist, or an Auditor defined in Section 2.2.
[3] That is a Trainee defined in Section 2.2.

regular users. They only need to know how to execute predefined commands and how to use the created lab environment. An intermediate definition contains all the information needed to build a working virtual lab. A user can supply these files as input to Sandbox Runner, which instantiates the actual virtual lab.

Sandbox Runner is a wrapper that incorporates a virtualization orchestrator and a provisioner. The orchestrator downloads the necessary OS images from an online or private repository and creates a *sandbox instance*, which contains the requested virtual machines and networks. It uses a type 2 virtualization hypervisor so the lab can be set up at a commercial-off-the-shelf host running a standard operating system.

The resulting sandbox instance may include one additional router (besides the defined machines), the so-called *border router*. The border router ensures the connection between different virtual networks and provides access to the Internet for all host machines. It also gives an opportunity to observe, manage, or analyze the traffic between the virtual networks, or their connection to the Internet. The provisioner handles the rest of the machine configuration and software installation. The created virtual machines can also send logs to a dedicated logging server, which can be hosted on the same host as the lab or the public Internet. This server can be configured the same way as any other machines in the virtual lab. Finally, after the virtual lab is created, a user can connect to a Capture the Flag (CTF) portal with an educational game and begin the training. The default configuration of a CTF portal will be provided, along with the configuration of the logging server. This will greatly simplify the setup and save the time and effort of educators. They will only need to modify aspects specific for their training.

Figure 3 depicts phases of how superusers and users interact with Cyber Sandbox Creator, Sandbox Runner, and the resulting virtual lab environment. The following subsections detail each stage. We expect that the typical users of the lab lack advanced cybersecurity or system administration skills (depicted as a *user* in Figure 3). They will run the Sandbox runner with input files provided by a *superuser* (see the definition above).

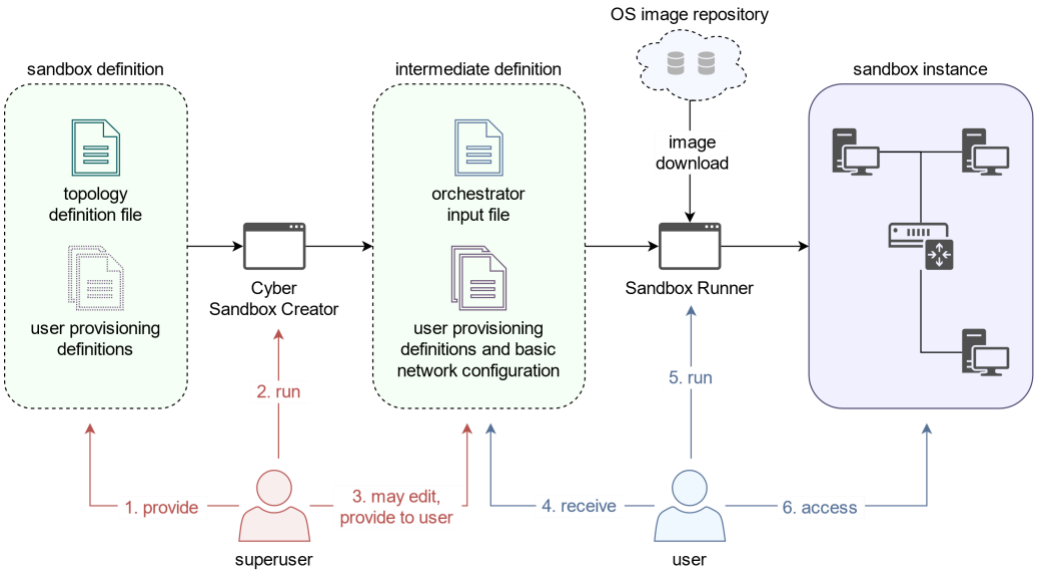

Figure 3: Virtual lab creation and its phases

5.1 Lab Environment Definitions

As an input to Cyber Sandbox Creator, we use two types of files denoted as *topology definition file* and *user provisioning definitions* in Figure 3. The first type is a YAML definition of the virtual machine parameters, the required OS images, and the network topology (aka *topology definition*). It uses a simple, human-readable format [Topology]. A superuser can add a second type of files for provisioning the selected virtual machines. These files are later used as input to the provisioner for the given virtual device. They define the desired configuration of virtual machines after their creation (including software installation or command execution).

The topology definition file is portable, easy to write, and powerful enough to describe even complex networks or requirements on virtual machines. The superuser can define hosts using a mandatory name and OS box parameters and a wide range of optional parameters, such as the number of virtual CPUs and operation memory to be used. Networks are defined by a name and an IP address in CIDR notation. The virtual machines are then assigned to networks in *net mappings*, which bind a hostname with a network name and give the host an IP address in the network. Router definitions and mappings are handled the same way, except that an OS image is automatically assigned to all routers. The example definition of a small lab with two networks, each with one host, is depicted in Figure 4.

```yaml
name: small-sandbox
hosts:
  - name: server
    base_box:
      image: ubuntu/xenial64
      man_user: vagrant
    flavor: csirtmu.tiny1x4

  - name: home
    base_box:
      image: ubuntu/xenial64
      man_user: vagrant
    flavor: csirtmu.tiny1x4

routers:
  - name: router
    cidr: 100.100.100.0/29

networks:
  - name: server-switch
    cidr: 192.168.20.0/24

  - name: home-switch
    cidr: 192.168.30.0/24

net_mappings:
  - host: server
    network: server-switch
    ip: 192.168.20.5
  - host: home
    network: home-switch
    ip: 192.168.30.5

router_mappings:
  - router: router
    network: server-switch
    ip: 192.168.20.1
  - router: router
    network: home-switch
    ip: 192.168.30.1
```

Figure 4: Topology definition file

5.2 Cyber Sandbox Creator implementation and output

Cyber Sandbox Creator is implemented as a Python script that converts and validates the input YAML topology definition and provisioning definitions to another format. The output (*intermediate definition*) serves as an input for the virtualization orchestrator and provisioner. It parses the input file using the PyYAML framework [PyYAML]. Then, it adds several more definitions, including the border router. The output is generated using the template engine Jinja [Jinja]. It is a widely used tool, which makes the output generation cleaner and more manageable. It also integrates well with the other tools we use.

Cyber Sandbox Creator produces a directory (Figure 5) as an output with an intermediate definition. Inside this directory is an input file for the virtualization orchestrator and two or three subdirectories for the provisioner. One contains the network configuration, the second user configuration for the provisioner, and the third may contain additional user files such as files with variables for the provisioner.

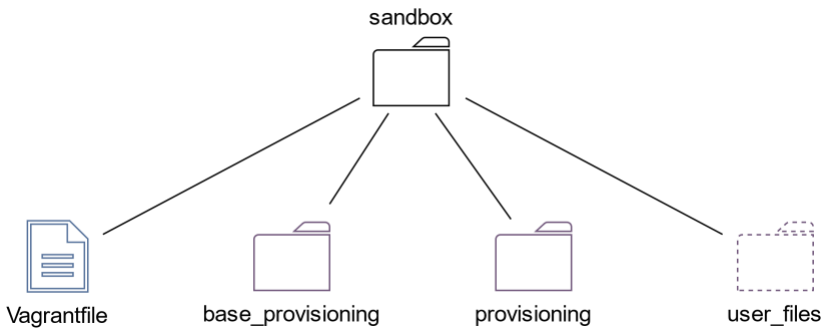

Figure 5: Structure of an intermediate definition

The input of the orchestrator is a single text file that contains all definitions of virtual machines and networks. These definitions are still structured and human-readable, but more complicated so that only expert users can easily edit them.

The first generated directory contains provisioner files for configuring the routing between the networks after the virtual machines are created.

The second directory is generated for users. It contains empty provisioner files for all defined virtual machines separately, one file for all hosts, and one for all routers. They are separated from the network definitions and structured in a way that adding new commands should be understandable even for non-expert users. If the superuser provided additional files for provisioning, they are copied to this directory, and the example files are not generated.

The third directory is present only if the superuser has provided another external configuration files during the creation. These files are copied in this directory to ensure the portability of the sandbox.

5.3 Sandbox Runner implementation and output

Sandbox Runner is a wrapper that runs the virtualization orchestrator and the provisioner. Together, these two tools create an instance of the virtual lab inside the type 2 virtualization hypervisor.

As the type 2 hypervisor, we chose VirtualBox since it is free on all three supported operating systems (in contrast to VMware Workstation Player, for instance). We chose Vagrant as the virtualization orchestrator because it is more lightweight than OpenStack and usable on a single computer. Ansible was chosen for provisioning because it works without agents on every host machine. Vagrant supports calling several provisioners, including Ansible after a virtual machine is created.

Vagrant handles the images of operating systems automatically. Pre-configured images that Vagrant uses are called boxes. Vagrant Cloud [VagrantCloud] is a public repository of these boxes, and it contains different versions of various operating systems. Vagrant can also work with boxes that are not present in the cloud. We provide some custom boxes that are configured and tested with Sandbox Runner. These boxes have fewer issues with compatibility, and they are more secure to use than unknown public boxes.

Sandbox Runner allows us to set additional options before running Vagrant, such as syntax check of the Vagrantfile, choosing Vagrant input file, or the provisioning method. It can check if there are enough computational resources at the host computer for the virtual machines before building them. If not, it can suggest instantiation only of a part of it. It can also make use of basic Vagrant commands to turn off virtual machines of the *sandbox instance* or suspend and recover them similarly to the sleep mode of any PC. Sandbox Runner takes the *intermediate definition* and passes it to Vagrant. Vagrant downloads the defined OS images (base boxes) from Vagrant cloud, a public repository for OS images for Vagrant, and builds the specified virtual machines and networks. Then, it calls Ansible, which configures the created devices using the provided files to their desired state. When the Sandbox Runner is finished, the *sandbox instance* is ready to use.

5.4 Access to the Created Lab Environment

The lab environment is created inside VirtualBox using Vagrant. Hence, it can be accessed by the means supported by setup. The ready-made way of access is through SSH. Since Vagrant handles the authentication, a user can access a virtual machine using the command `vagrant ssh <hostname>` and does not need to bother with SSH key management. All the created devices, including routers, can be accessed this way. We also plan to support access to a graphical user interface of virtual hosts via the Remote Desktop Protocol, which is supported by VirtualBox.

If the lab is used for any activities featuring a CTF portal or logging server as depicted in Figure 2, the respective access channel at the host computer is used (web browser or SSH client).

6 Demonstration Cases of the Lab Environment

This section presents examples of three specific and realistic use cases of the lab: testing, education, and certification.

6.1 Developing a Testing Environment

Suppose a Researcher, Developer, or a Specialist wants to create a virtual environment suitable for their needs. The lab allows defining the properties of the virtual machines (such as their operating system and allocated computational resources), their networking, and configuration (such as the installed software and content of their file system). The first two aspects will be covered by the output of the Cyber Sandbox Creator. The custom configuration of created virtual machines remains the only responsibility of the user. The benefit of CSC is that the generated lab environment is replicable and easily maintained. In the long run, it is more usable and robust than the standard approach of generating and importing ISO machine images or snapshots of virtual hosts. Since the definitions are portable, users can benefit from mutually sharing the already created definitions or choose any of the definitions publicly available from the Ansible Galaxy repository [AnsibleGalaxy].

The testing environment will be sandboxed to allow safe and free experimentation. It will run on an arbitrary operating system as long as the usage requirements are satisfied. In its most basic use case, CSC acts as a wrapper around Vagrant. However, it has numerous advantages compared to writing a Vagrantfile from scratch. CSC simplifies the topology specification, especially the networking definition, which is time-consuming to set up manually. Vagrant Cloud [VagrantCloud] offers a large variety of pre-built images of operating systems, which range from base boxes to full application stacks or development environments. We provide several reliable and tested operating systems boxes via the Vagrant Cloud, along with a comprehensive documentation for installation, usage, and troubleshooting. In contrast to unknown boxes which are often undocumented, the provided boxes are tested and optimized for using with CSC.

6.1.1 Local development of sandboxes for KYPO Cyber Range Platform

In coordination with CONCORDIA [CONCORDIA], another Cyber Security Network of Competence Centre, we provide a cross-platform and cross-project compatibility showcase in the area of cyber range development. Sandbox definitions used by Cyber Sandbox Creator are compatible with the sandbox definition files for KYPO cyber range platform [Topology]. One can develop and test new sandboxes locally before building them in the cloud-based KYPO platform. This approach simplifies and speeds up the creation of new definitions and their testing. Sandbox development in KYPO is often more time-consuming than the local testing, so doing most of the work locally saves time compared to debugging at the KYPO platform.

6.2 Using the Lab in Educational Activities

An Educator can use the virtual environment (described in the previous section) in hands-on training. Each Trainee will receive an identical copy of the virtual environment that is isolated from other Trainees, as shown in Figure 6. This allows the trainees to practice cyber attacks and defense measures without any negative consequences on physical systems. The Trainees do not need advanced cybersecurity skills to run the training environment at their hosts.

This use case additionally requires a shared portal for presenting the assignments of tasks to complete in the virtual environment. It can also include scoring, hints, submissions of solutions, and other features relevant to the training. We will consider using picoCTF or CTFd described in Section 4.5. In the later version, the basic steps of the integration with the CTF portal will be ready to save the time and effort of educators.

This setup includes local lab environments run at own hosts of individual trainees, along with one central node with the shared portal that must be reachable from all local lab environments. As a result, the requirement for computational resources that accommodate educational activities of trainees is delegated to individual participants. Running the lab environment locally means that trainees can access environment definitions (such as configuration of installed software or an intentional vulnerability), which might be unsuitable in educational contexts such as summative assessment of trainees.

Since the lab will support logging of various events in the created environment, educators can monitor the progress of trainees during an ongoing activity at a central Log Server (see Figure 6). This monitoring is important for teachers and educational researchers to understand the trainees' learning processes, assess them, and provide personalized feedback. Logs need to capture the trainee's activities in the lab, such as the submitted shell commands, in a human-readable, structured format. At the same time, they should not store any personally identifiable information to preserve the privacy of the trainees. Moreover, it is crucial that the transfer of the logs from the trainee to the Log Server is encrypted and authenticated. Storing and processing the logs can be done in ELK (described in Section 4.4) and should offer the option for real-time handling during the training.

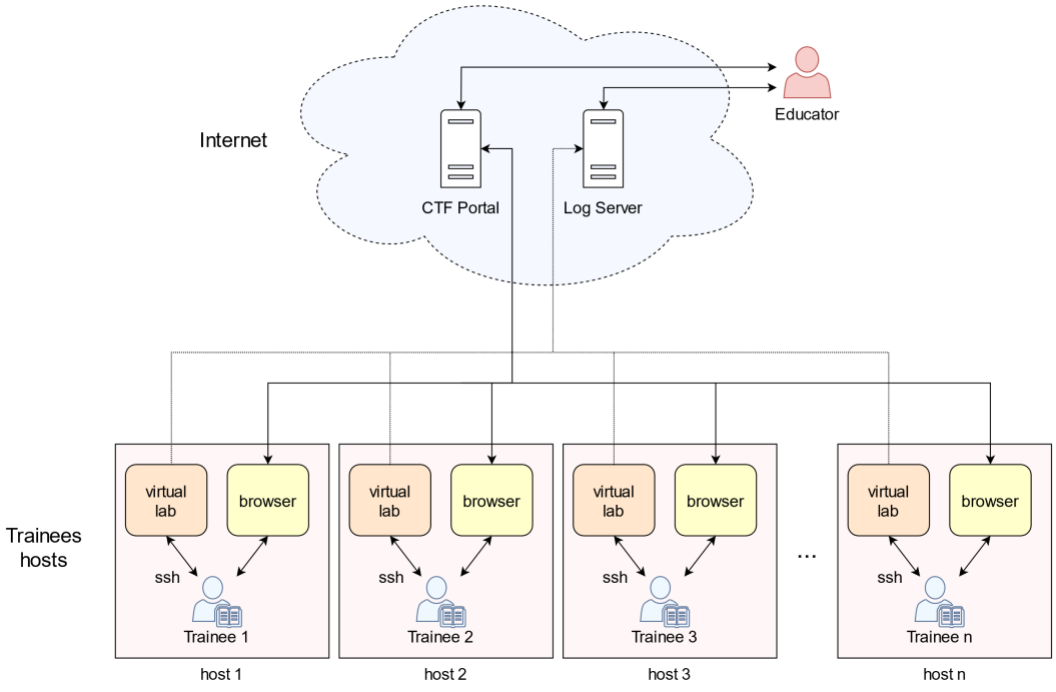

Figure 6: Multiple virtual labs in educational activities

6.3 Using the Lab for Certification Activities

An Auditor can use the lab environment to validate the features of HW equipment and SW tools and test their characteristics. Initially, the Auditor may want to develop a custom environment that will then be used repeatedly. The testing environment will be instantiated automatically and provides the same conditions for every test run. This is important for the reproducibility of certification activities in the future, not only by the Auditor, but everyone else who might want to verify the compliance of the certified HW and SW. At the same time, a lab environment build by CSC is limited by the capabilities and robustness of the CSC components: virtualization hypervisor and orchestrator. If these components fail to ensure these features, CSC has very limited options to do so.

The environment will be able to access HW connected via USB ports of the computer hosting the lab (see Figure 7). The tested SW will be installed and configured at the created virtual hosts. If the used SW can produce any logs in the standard format (Syslog), the lab will support their collection and analysis using a local or remote instance of logging server and analytical tool, such as ELK described in Section 4.4.

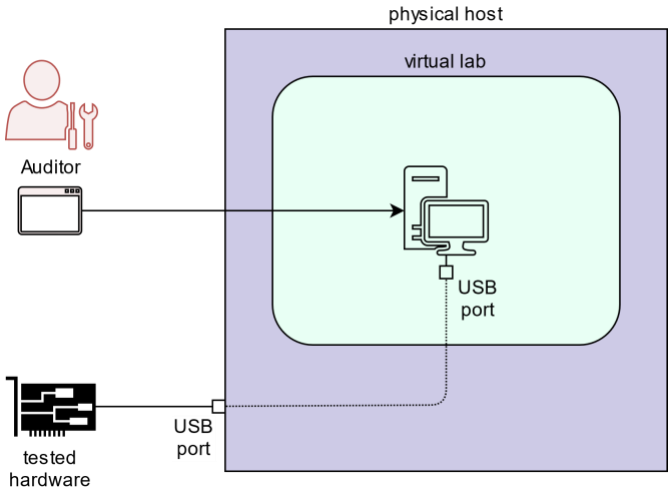

Figure 7: Virtual lab for hardware testing

7 Conclusion

This report presents the architecture of an open-source, lightweight, virtual lab environment for cybersecurity education, testing, and certification. The lab environment relies on several proven components that enable automated creation and instantiation of a virtual network of several hosts at a local, commercial off-the-shelf computer. Given this constraint, we analyzed user requirements, reviewed available building blocks, and chose suitable technologies to support all three main cybersecurity use cases. This report is one part of the D7.2 deliverable; the other is the source code and documentation of the first version of the tool for building and instantiating the lab environment named Cyber Sandbox Creator (CSC in short) available at https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator.

7.1 Current status

So far, we developed a prototype ready for testing and basic usage. As of the end of 2020, CSC can create an intermediate definition of a sandbox with Linux virtual machines that can be built and run on Linux, MS Windows, and MacOS hosts. A superuser can add or import additional Ansible playbooks to configure the sandbox or pass extra variables to Ansible. A definition can be created with or without a border network router.

We successfully tested the connection between the virtual lab and a hardware device plugged in the USB port of a host machine. This was the first step towards the support of the certification use case outlined in Section 6.3.

Sandbox Runner does not exist, yet. Currently, sandbox instances are built using Vagrant directly.

Regarding custom images of virtual machines, we prepared and published our own Kali Linux image, which is available on the Vagrant cloud [MuniKali]. This image was tested and works well with CSC.

We have tested CSC at Masaryk University, Brno, Czech Republic, to prepare and configure custom environments for cybersecurity training of our students. Students were provided with the created lab and completed their tasks there or worked on their thesis projects. We have recently provided CSC to instructors at Slovak Technical University in Bratislava, who will follow this approach. We also used CSC for local development of sandboxes for the KYPO Cyber Range Platform [KYPO-CRP] in cooperation with the CONCORDIA pilot. Finally, CSC is also now being tested in another use case by a Czech company developing Manufacturing Execution Systems.

To sum up, the requirements for the lab presented in Section 2.2 are now met as follows:

1. Lab must be easy to use. Instantiation and disposal of the lab environment must be as automated as possible (ideally in one click). The lab must prevent its users from making common mistakes. [easy to use]: *Partially implemented. Validation and checks of the user input are not implemented yet.*
2. Lab environment is built with minimal overhead so that it can be run at a single commercial off-the-shelf computer. [minimal system resources required]: *Implemented.*
3. Lab environment is built using free and open-source components, no additional fee is required for using any component at common operating system families (Windows, Linux, macOS). [free and open-source components]: *Implemented.*
4. Lab enables users to run hosts with operating systems different to the operating system of the host computer. [interoperability]: *Partially implemented. Linux OS can be run in the lab hosted at Windows OS. The support of Windows OS in the lab is not implemented yet.*

5. Lab supports running a distributed training session, i.e., each trainee working at their own host with the lab. [distributed training]: *Not implemented yet.*
6. Lab enables access to hardware connected to the host at which the lab is running. [access to hardware]: *Not implemented yet.*

Properties of the virtual environment instantiated using the lab:

7. Lab instantiates networks of full-fledged virtual machines, i.e., it emulates the networked hosts that run a standard operating system. [full OS emulation]: *Implemented.*
8. Lab allows controlling the level of isolation of the created environment from outside. The environment can be completely isolated as well as connected to the Internet. [isolation] *Not implemented yet. However, by default, hosts in the lab are not reachable from the Internet.*
9. Lab can be suspended and its internal state saved. It can be later recovered to its previous state. [suspendability]: *Not implemented yet.*
10. Lab supports logging of various events (such as commands typed by learners or researchers), either locally or at a remote central endpoint, so that users can track the ongoing and past processes in the lab environment. [logging]: *Not implemented yet.*

Using the lab to define a virtual network:

11. The definition of the virtual network and machines must be described in structured formats that allow simple modifications according to various and dynamic user needs. [infrastructure as code]: *Implemented.*
12. The definition of the virtual network and machines must be well-documented and ready for sharing with others who would like to instantiate the same environment, either in the lab or in a full-fledged cyber range. [portability]: *Partially implemented. Definitions can be shared, more documentation is needed.*

7.2 Plans for further development

Regarding the lab infrastructure, we plan multiple improvements that are currently a work in progress:

- Creating more images of virtual hosts (base boxes with standard operating systems and toolsets). Since the publicly available boxes usually do not include source definitions, using an unknown image without knowing its contents may impair the security and robustness of the built lab. We will especially focus on adding support for MS Windows operating system boxes.
- Developing unit tests and maintaining integration tests for the CSC.
- Estimating the required operating memory and other requirements for the intermediate definitions. This will help the Sandbox runner to check whether enough system resources are available before the lab building process starts.
- Supporting the use cases of multiple virtual labs. This will allow several labs to run in parallel, as indicated in Figure 6.
- Selecting a suitable CTF portal and providing an example configuration that can be deployed at a remote server, along with an example of the log server configuration.

Regarding the lab applications, there are several ongoing projects. As mentioned above, we sent the current version of the CSC to various early adopters. They will use it for teaching cybersecurity classes, especially for creating and hosting cybersecurity training sessions and serious games. To support this use case, we are proposing structured guidelines for creating cybersecurity serious games [SIGCSE2021] along with an

example game that demonstrates the capabilities of the CSC. Finally, we plan to test the federation of the lab with JAMK RGCE cyber range [RGCE]. JAMK proposed two types of federation [D7.1] allowing CSC to be used in two main ways.

- In the technical federation, labs and cyber ranges are interconnected to share computational resources. This can prove useful during a large cyber exercise when different teams can distribute their work among different environments. CSC can emulate a remote lab in the exercise scenario.
- In the operational federation, labs and cyber ranges share data, such as activity logs. This is especially beneficial to foster research and development in cybersecurity. Gathering research data from cybersecurity contexts is usually difficult and time-consuming. However, when several labs federate to participate in the data collection, they can accumulate a larger and more diverse dataset that is representative of various use cases and user populations.

References

[Ansible] Red Hat. Ansible is Simple IT Automation. Online at https://www.ansible.com/ [Accessed on December 16, 2020].

[AnsibleGalaxy] Galaxy. Online at https://galaxy.ansible.com/ [Accessed on December 16, 2020].

[Avatao] Avatao. Learn to build secure software. Online at https://avatao.com/ [Accessed on December 16, 2020].

[Chapman14] Peter Chapman, Jonathan Burket, and David Brumley. PicoCTF: A Game-Based Computer Security Competition for High School Students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. 2014. https://www.usenix.org/conference/3gse14/summit-program/presentation/chapman

[Chef] Chef. Compliance Auditing. Online at https://www.chef.io/ [Accessed on December 16, 2020].

[Chung17] Kevin Chung. Live Lesson: Lowering the Barriers to Capture The Flag Administration and Participation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*. 2017. https://www.usenix.org/conference/ase17/workshop-program/presentation/chung

[Cloudbuilder] KVM and QEMU – do you know the connection? Online at https://cloudbuilder.in/blogs/2014/03/09/kvm-and-qemu/ [Accessed on December 16, 2020].

[coldcorectf] ColdCore. Online at https://github.com/IceCTF/ColdCore [Accessed on December 16, 2020].

[Comparitech] 13 Best Log Management & Analysis Tools. Online at https://www.comparitech.com/net-admin/log-management-tools/ [Accessed on December 16, 2020].

[CONCORDIA] CONCORDIA. Cyber security cOmpeteNce fOr Research anD Innovation website. Online at https://www.concordia-h2020.eu [Accessed on December 16, 2020].

[Cyris] CyRIS: Cyber Range Instantiation System. Online at https://github.com/crond-jaist/cyris [Accessed on December 16, 2020].

[D7.1] Cyber Security for Europe. Report on existing cyber ranges, requirements. 2020. Online at https://cybersec4europe.eu/wp-content/uploads/2020/09/D7.1-Report-on-existing-cyber-ranges-and-requirement-specification-for-federated-cyber-ranges-v1.0_submitted.pdf [Accessed on December 16, 2020].

[Docker] Docker. Empowering App Development for Developers. Online at https://www.docker.com/ [Accessed on December 16, 2020].

[ECSO] European Cyber Security Organisation. 2020. Understanding Cyber Ranges: From Hype to Reality. Technical Report. (ECSO). https://ecs-org.eu/documents/publications/5ea47a760f24f.pdf [Accessed on December 16, 2020].

[Elasticsearch] Elasticsearch. Online at https://github.com/elastic/elasticsearch [Accessed on December 16, 2020].

[ElasticStories] Use Cases - Elastic Stack Success Stories. Online at https://www.elastic.co/customers/ [Accessed on December 16, 2020].

[ELK] ELK Stack. Online at https://www.elastic.co/what-is/elk-stack [Accessed on December 16, 2020].

[ENISA] The European Union Agency for Cybersecurity (ENISA). EU cybersecurity certification framework. 2019. Online at https://www.enisa.europa.eu/topics/standards/certification [Accessed on December 16, 2020].

[EUROPA19] European Commission, Digital Skills at the core of the new Skills Agenda for Europe. 2016. Online at https://ec.europa.eu/digital-single-market/en/news/digital-skills-core-new-skills-agenda-europe [Accessed on December 16, 2020].

[Fluentd] Open Source Data Collector. Online at https://www.fluentd.org/ [Accessed on December 16, 2020].

[FluentdTestimonials] Testimonials. Online at https://www.fluentd.org/testimonials [Accessed on December 16, 2020].

[Guardtime] Guardtime. Guardtime cyber. 2018. Online at https://cyber.guardtime.com/exercises/ [Accessed on December 16, 2020].

[Irvine17] Cynthia E. Irvine, Michael F. Thompson, Michael McCarrin, and Jean Khosalim. Live Lesson: Labtainers: A Docker-based Framework for Cybersecurity Labs. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*. 2017. https://www.usenix.org/conference/ase17/workshop-program/presentation/irvine

[ISC2] (ISC)², Cybersecurity Workforce Study. 2019. Online at https://www.isc2.org/Research/2019-Cybersecurity-Workforce-Study [Accessed on December 16, 2020].

[Jinja] Jinja. Online at https://jinja.palletsprojects.com/en/2.10.x/ [Accessed on December 16, 2020].

[JTF] The Joint Task Force on Cybersecurity Education (JTF). ACM/IEEE/AIS SIGSEC/IFIP Cybersecurity Curricular Guideline. 2017. Online at https://cybered.acm.org [Accessed on December 16, 2020].

[Kibana] Kibana. Online at https://github.com/elastic/kibana [Accessed on December 16, 2020].

[Kubernetes] Kubernetes. Online at https://kubernetes.io/ [Accessed on December 16, 2020].

[Kucek19] Stela Kucek, Maria Leitner: An Empirical Survey of Functions and Configurations of Open-Source Capture the Flag (CTF) Environments. Journal of Network and Computer Applications, Volume 151, 2020. https://doi.org/10.1016/j.jnca.2019.102470.

[KVM] KVM. Kernel Virtual Machine. Online at https://www.linux-kvm.org/page/Main_Page [Accessed on December 16, 2020].

[KYPO] Jan Vykopal, Radek Ošlejšek, Pavel Celeda, Martin Vizváry, Daniel Tovarňák. KYPO Cyber Range: Design and Use Cases. In Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT. Madrid, Spain: SciTePress, 2017. pp. 310-321, 12 p. ISBN 978-989-758-262-2. doi:10.5220/0006428203100321.

[KYPO-CRP] Masaryk University. KYPO Cyber Range Platform. Online at https://crp.kypo.muni.cz/ [Accessed on December 16, 2020].

[Lability] Lability. Online at https://github.com/VirtualEngine/Lability [Accessed on December 16, 2020].

[Logstash] Logstash. Online at https://github.com/elastic/logstash [Accessed on December 16, 2020].

[MuniKali] munikypo/kali-2019.4. Online at https://app.vagrantup.com/munikypo/boxes/kali-2019.4 [Accessed on December 16, 2020]

[NIST] National Institute of Standards and Technology. Cyber Ranges. 2018. Online at https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf [Accessed on December 16, 2020].

[OpenStack] Openstack. Online at https://www.openstack.org/ [Accessed on December 16, 2020].

[OpenStackDocs] OpenStack Documentation. Online at https://docs.openstack.org/newton/install-guide-rdo/overview.html#figure-hwreqs [Accessed on December 16, 2020].

[picoctf] picoCTF. Online at https://github.com/picoCTF/picoCTF [Accessed on December 16, 2020].

[picoctfgame] picoCTF. Online at https://picoctf.com/ [Accessed on December 16, 2020].

[Puppet] Puppet. Stop talking about technology. Start delivering value. Online at https://puppet.com/ [Accessed on December 16, 2020].

[PyYAML] Welcome to PyYAML. Online at https://pyyaml.org/ [Accessed on December 16, 2020].

[QEMU] QEMU the FAST! Processor emulator. Online at https://www.qemu.org/ [Accessed on December 16, 2020].

[RGCE] JAMK. Realistic Global Cyber Environment. Online at https://jyvsectec.fi/wp-content/uploads/2018/10/JYVSECTEC-cyber-range.pdf [Accessed on December 16, 2020].

[RHEA] RHEA Group. Cybersecurity training and simulation. Online at http://products.rheagroup.com/cybersecurity-training [Accessed on December 16, 2020].

[SIGCSE2021] Miriam Gáliková, Valdemar Švábenský, and Jan Vykopal. 2021. Toward Guidelines for Designing Cybersecurity Serious Games. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21) (To appear)*. Association for Computing Machinery, New York, NY, USA, 561. ISBN 978-1-4503-8062-1. doi: 10.1145/3408877.3439568.

[shadowsoft] 3 Open Source Container Management Tools (Comparison and Review). Online at https://shadow-soft.com/open-source-container-management-tools/ [Accessed on December 16, 2020].

[Taylor17] Clark Taylor, Lawrence Livermore, Pablo Arias, Jim Klopchic, Celeste Matarazzo, and Evi Dube. CTF: State-of-the-Art and Building the Next Generation. In *2017 USENIX Workshop on Advances in Security Education (ASE 17)*. 2017. https://www.usenix.org/conference/ase17/workshop-program/presentation/taylor

[Topology] Masaryk University. KYPO Cyber Range Platform Topology Definition. 2020. https://doi.org/10.5281/zenodo.4314736

[Technologyadvice1] VMware vs. VirtualBox: Which is Better for Desktop Virtualization? Online at https://technologyadvice.com/blog/information-technology/vmware-vs-virtualbox/ [Accessed on December 16, 2020].

[Techradar] Best virtual machine software of 2020: virtualization for different OS. Online at https://www.techradar.com/best/best-virtual-machine-software [Accessed on December 16, 2020].

[tjctf] TJCTF. Online at https://github.com/TJCSec/ctf-platform [Accessed on December 16, 2020].

[Upguard1] Ansible vs Puppet. Online at https://www.upguard.com/articles/ansible-puppet [Accessed on December 16, 2020].

[Upguard2] Ansible vs Chef Updated for 2019. Online at https://www.upguard.com/articles/ansible-vs-chef [Accessed on December 16, 2020].

[Vagrant] Vagrant. Development environments made easy. Online at https://www.vagrantup.com/ [Accessed on December 16, 2020].

[VagrantCloud] Vagrant Cloud. Online at https://app.vagrantup.com/boxes/search [Accessed on December 16, 2020].

[VirtualBox] VirtualBox. Online at https://www.virtualbox.org/ [Accessed on December 16, 2020].

[VMwarePlayer] VMware Workstation Player. Online at https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html [Accessed on December 16, 2020].

[Yamin20] Muhammad Mudassar Yamin, Basel Katt, Vasileios Gkioulos. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. In *Computers & Security, Volume 88*. 2020. https://www.sciencedirect.com/science/article/pii/S0167404819301804

[yuktictf] Yukti CTF. Online at https://github.com/chirathr/YuktiCTF [Accessed on December 16, 2020].