On Automatic Cyber Range Instantiation for Facilitating Security Training

By Pham Duy Cuong

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Razvan Beuran

March, 2017

On Automatic Cyber Range Instantiation for Facilitating Security Training

By Pham Duy Cuong (1510047)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Razvan Beuran

and approved by
Professor Mizuhito Ogawa
Professor Yoichi Shinoda
Associate Professor Ken-ichi Chinen

February, 2017 (Submitted)

Abstract

As cyber attacks are happening worldwide nowadays, cybersecurity training and education become highly important. In modern training programs, controlled training environments, so-called cyber ranges, appear as an efficient way for trainees to gain practical knowledge through hands-on activities. However, creating an environment that contains all the necessary features and settings, such as virtual machines, network topology and security-related content, is not an easy task, especially for a large number of participants.

This thesis presents CyRIS (Cyber Range Instantiation System), which is an open-source tool for facilitating cyber range creation. CyRIS provides a mechanism to automatically prepare and manage cyber ranges for cybersecurity training based on specifications defined by instructors. It contains both basic functions for setting up the infrastructure of the cyber range, and security-related functions, which can be used to reproduce actual incidents for the purpose of creating security content. This group of functions is the main difference between CyRIS and other well-known automated environment configuration tools, as it greatly reduces instructors' time and effort in constructing a realistic training environment.

In this thesis, we first describe the design and implementation of CyRIS, as well as its utilization. We then present an evaluation of CyRIS in terms of feature coverage compared to the "Technical Guide to Information Security Testing and Assessment" of the U.S. National Institute of Standards and Technology, and in terms of functionality compared to other similar tools. We also discuss the execution performance of CyRIS, both on its own in regard to the running time for constructing representative cyber ranges with large-scale scenarios, as well as in comparison with Alfons, which is a recent tool in the field and developed by Japan National Institute of Information and Communications Technology.

Keywords: Cybersecurity, cybersecurity training and education, cyber range, training environment, network security.

Declaration: I hereby declare that this whole dissertation is my own work, and that it has not been previously included in any other thesis, dissertation or report.
Student: Pham Duy Cuong

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Associate Professor Razvan Beuran, School of Information Science, Area of Security Network, JAIST, for his extraordinary support of my master, for his patience, motivation and immense knowledge. His guidance has always been my inspiration and has helped me overcome numerous obstacles I have been facing throughout my research. Without his persistent support this thesis would not have been possible.

I would like to convey my special thank to Professor Xavier Defago, Department of Computer Science, School of Computing, Tokyo Institute of Technology, as he was my supervisor in the first year of my master, for his financial support, his insightful advices and encouragement, not only in research but also in other aspects of life. My sincere thanks are also due to Professor Yoichi Shinoda and Associate Professor Ken-ichi Chinen, School of Information Science, Area of Security Network, JAIST, as committee members of my midterm and final defenses, who provided instructive questions and helpful comments to clarify and improve my research.

I would also like to thank Mr. Tang Thanh Dat, Mr. Bui Ha Duong and Mr. Ngo Van Thuan, my master fellows at JAIST, as I am gratefully indebted to them for their valuable comments and support during the time I conducted my research.

Finally, I owe my loving thanks to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and my life in general. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cybersecurity, also known as IT security, focuses on protecting networks, computers, programs and data from attack, damage, unintended or unauthorized access. Currently cyber-attacks are happening worldwide on a daily basis. In 2015, hackers breached the systems of the health insurer Anthem Inc., the U.S., exposing nearly 80 millions personal records with information of birthdays, addresses, social security numbers, and emails [1]. Also in the same year, the Japan Pension Service system was hacked, leading to 1.25 millions cases of personal data being leaked [2]. The method used in this attack was a classical one: a targeted email attack, in which a virus-laced email attachment disguised as a health ministry document was sent to and opened by employees. The same trend has been observed in 2016: OVH, one of the world's top hosting companies, reported in September that its systems were hit by distributed denial-of-service (DDoS) attacks that nearly reached one terabit per second (Tbps) [3]. This was considered as the largest DDoS offensive ever seen so far on the Internet. Another incident recently happened in Vietnam, when flight display screens at the country's two largest airports were hacked to show messages criticizing Vietnam's claims of territory in the South China Sea [4].

In this context, cybersecurity education and training appear to be more and more critical, as one of the best ways in which such cyber breaches can be prevented and handled adequately. As a result, there are many available training programs nowadays. SANS Institute [27], a trusted and by far the largest source for information security training in the world, provides many training programs in the cybersecurity field, both as live and online courses. Udemy, which is a famous online learning platform, offers a paid course called "Cyber Security" [5] for introducing a wide-range overview of cybersecurity concepts and practices, including threat analysis and risk management, encryption, firewalls, and intrusion detection.

In Japan, CYDER [8] is a cybersecurity program coordinated by the Ministry of Internal Affairs and Communications. Its purpose is to provide hands-on training to IT personnel of government organizations and top companies. Hardening Project [7], on the other hand, is a security contest organized by the Web Application Security Forum. A number of teams, consisting of security experts and IT professionals, compete with each other in respect of the security improvements they can provide in a realistic e-commerce company-like network.

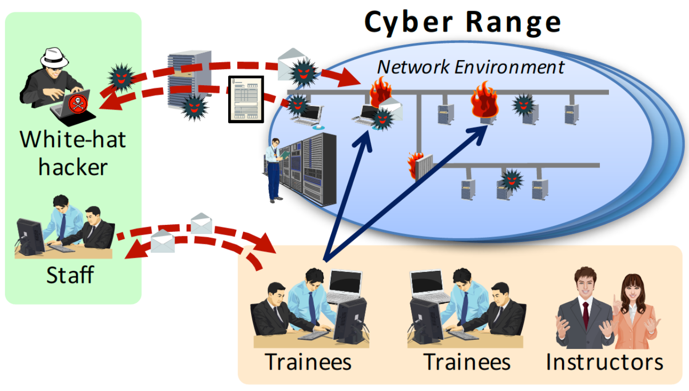


Figure 1.1: Overview of a cybersecurity training program using a cyber range for hands-on activities [12].

In these cybersecurity training programs, most of the content requires participants to deal with challenging and hands-on problems in real world scenarios. A cyber range is a training environment that the participants can access and investigate to find answers to questions, and to acquire practical skills (Figure 1.1). It is specifically designed for cybersecurity training, as it contains all the required infrastructure (machines, networks, tools, etc.) and security settings (vulnerability, flaws, trace, etc.). The cyber range also needs to be well controlled in a networking perspective: isolated from the outside to avoid traffic leakage, and separated among trainees to prevent them from accessing each other's environment.

However, many of the current training programs rely on manual setup and configuration of the training environments, which is time consuming, lots of efforts are required, and is error-prone. Although some training programs might make use of specific tools to facilitate the preparation task, such tools are undisclosed, therefore they do not benefit the society at large. Moreover, a recent analysis made by Beuran et al. about the best practices and methodologies in cybersecurity training [11], which surveys and presents in detail major training programs in Japan, has shown that only one of them, the Hardening Project, is currently employing tools for automating the setup task. The mentioned automation, however, refers only to the environment construction, and not to the security content creation inside, which is still done manually.

Being aware of these issues, in this thesis, we propose CyRIS [13] (Cyber Range Instantiation System), as a solution for reliably and repeatably creating and managing cybersecurity training environments. CyRIS is a component of CyTrONE (Cybersecurity Training and Operation Network Environment), which is being developed at JAIST (Japan Advanced Institute of Science and Technology), as an integrated framework for automating the content generation and environment setup tasks for cybersecurity training programs [12].

In CyRIS, a cyber range description is created by the instructors to specify in detail the cyber range for the training. This description, on one hand, provides the instructors with a means to thoroughly review the security content and the environment specifications which are going to be used in their course. The description, on the other hand, serves as input for CyRIS to automatically create the desired environment without human interaction. CyRIS offers both basic setup functions that are commonly found in other automated configuration management tools, and security-related functions, which can be used to reproduce actual incidents for the purpose of creating security content. This group of functions is unique in CyRIS compared to other tools, and they play a key role in creating realistic cyber ranges.

The main contributions of this thesis are as follows:

- The design and implementation of CyRIS as an automated open-source tool for cybersecurity training environment creation. CyRIS contains functions to help instructors prepare security-related content for the cyber range. Moreover, it offers a module for configuring a cyber range with arbitrary network topology and services in a convenient and flexible manner.

- The evaluation of CyRIS in terms of both feature coverage and performance. By using the U.S. NIST Technical Guide to Information Security Testing and Assessment as a reference [24], we show that CyRIS has functionality to create complex cyber range environments with sophisticated security-related content for cybersecurity training programs. The evaluation of CyRIS performance demonstrated that it meets the reasonable time for creating large-scale cyber range construction.

The remainder of this thesis discusses related work (Chapter 2), presents CyRIS in detail (Chapter 3), evaluates our system in several perspectives (Chapter 4), and ends with conclusions and future work (Chapter 5).

Chapter 2

Motivation

Various tools have been published that provide users an automated way to configure both physical and virtual environments for general purposes. In this chapter, we first define the functions that are required for a tool to be able to construct a complete cyber range environment. After that, we discuss in detail about well-known tools in both their advantages and drawbacks.

2.1 Functionality Requirements

As having said briefly in Chapter 1, a cyber range is a well-defined virtual environment used in a cybersecurity training program. After having surveyed and analyzed current available training programs, we found several characteristics that a cyber range should possess. It, first of all, needs to contain all the content and configuration needed for the program (machines, tools, vulnerability, network topology, etc.). Secondly, it should be well controlled regarding the networking perspective, in which it has to be isolated from the outside to prevent traffic leakage for the safety purpose, as well as separated among participants for the fairness purpose during the training. Given these conditions, an orchestrating tool, in order to be able to construct a complete cyber range, needs to satisfy several requirements described below:

- *Individual node content management.* A cyber range environment consists of a number of machines, each of them plays a different role in the network, and therefore contains a different content compared to the others. Moreover, various topics can be taught in cybersecurity training, such as forensics analysis, penetration testing, encryption, etc. The corresponding security-related content, such as attack traces, vulnerability, logs, etc., for each of these topics also vary. It is therefore essential for the orchestrating tool to have a means for creating those content automatically in a flexible and convenient manner, as that instructors of the programs no longer have to generate them manually.

- *Network service configuration.* A cyber range reflects a practical network, which consists various services and topology. For example, the network might have a complex topology that are divided into several segments, and a number of forwarding rules are needed for those segments to communicate with each other. The orchestrating tool, as a result, should be able to handle this by having a function for configuring the network services, again, in a flexible manner.

- *Environment isolation.* A cyber range is an environment for the participants to access and practice various technical skills during the training, including "dangerous" activities such as attacking. These activities, if not being controlled well, might potentially lead to nasty circumstances with serious consequences. Therefore, it is one of the most important requirements from the orchestrating tool to be able to isolate the environment from the outside network, as making sure that no traffic leakage could possibly happen. Moreover, as the fact that the participants might look over others' environments for copying answers, it is also necessary to apply some certain methods to separate and prevent them from doing so.

- *Open-source product.* Last but not least, it is important that the tool would be published as an open-source product. With the current status that cyber attacks are happening worldwide, cybersecurity training is now important not only in top organizations, companies, and in the military, but also to ordinary people who are interested in the topic and would like to defense themselves from such attacks. Having an orchestrating tool as an open-source product is one of the best solutions to have large-scale education and training programs that reach everybody in the society, including young people in universities, colleges, and even high schools.

2.2 Related Work

Currently, various tools have been published that provide users an automated way to configure the environment on an individual node, including installing and configuring applications, setting up the system, and so on. Ansible [14], Vagrant [16], and Chef [17] are well-known tools in this area. They allow users to specify settings and content in a "recipe" and create a desired environment from a clean node. However, none of these tools has the function of configuring network service and topology among nodes, and it also gets harder when users have to create multiple recipes for circumstances that involve setting up a large environment with many nodes.

For network management configuration tools, we have cloud controllers such as OpenStack [15] and VMWare vSphere [18]. These tools are able to create and manage a cloud system with multiple virtual machines and network service among them. Another tool is SpringOS [19] that is currently in use at StarBED facility [20] for managing physical nodes and network experiments. It has the functionality of controlling network topology among nodes, which is missing on the other tools. Moreover, it offers a function for installing an operating system in a physical machine from scratch. Nevertheless, these tools lack the ability of setting content and configuration on individual nodes. In addition, SpringOS is designed only for StarBED use purpose.

Shingo Yasuda et al. introduce Alfons [21] as a recent tool that can be used to create environments for cybersecurity training and malware analysis. In short, to the best of our knowledge, this is the tool which has open information and fits the context the most. They consider an original node as a clean operating system with unique data files inserted. Therefore, copying files and executing scripts are two main mechanisms Alfons uses to create a required environment in an individual node. Besides, by using SpringOS' API, Alfons has functions for creating various network topology in the cyber range, as well as for isolating it from the outside network. However, while the implementation appears simple and highly efficient, Alfons falls short in the purpose of helping instructors easily create security-related content in the cyber range. In other word, in order to use Alfons to create a desired environment, the instructors themselves have to somehow, and by some tools, generate in advance their content, such as an attack trace, a log file, etc, and copy them to the cyber range. This work is, again, boring and tedious. Besides, Alfons' source code is closed.

Recently, Facebook has introduced its own CTFs platform to open source [22]. Facebook's CTF provides users a free platform that takes care of the maps, team registration, and scoring. It offers a way to make security education easier and more accessible for people who are interested in information security and technical skills. It also brings schools, organizations, and others who lack resources a chance to host their competition and teach students and employees about hacking skills. On the contrary, it is limited in the range of security knowledge, when only a small set of challenges is publicly available in the competition and mainly focuses on hacking ability.

The SANS Information Security Institute [27] and CERT of the Software Engineering Institute [28] are two of the most famous and trustworthy organizations in the world regarding cybersecurity training. They offer hands-on, interactive and practical courses for a large number of trainees in different important topics about cybersecurity. However, their cyber range creation processes are not made public.

Compared to these tools, CyRIS is an open-source tool for creating cyber range environments in a flexible and efficient manner. It is implemented in the Python language. CyRIS has both functions for installing content on a large environment with many nodes, and for configuring the network service among them. Moreover, CyRIS offers built-in features for creating specific security-related content by launching real incidents, which greatly facilitates the preparation work for the instructors.

Chapter 3

Cyber Range Instantiation System

This chapter presents the design and implementation of the Cyber Range Instantiation System, so-called CyRIS, which is an open-source tool for automatic preparing and managing cyber ranges for cybersecurity training. The first section describes the overview of our approach, the second and the third goes into detail about the working flow and functionality, and the last section summarizes the characteristics of CyRIS.

3.1 Overview

CyRIS is the core component of the cybersecurity training framework CyTrONE [12], which is being developed at JAIST. The system architecture of CyTrONE is described in Figure 3.1. It consists of three main parts, which are:

- *Training Specification.* Based on user inputs and the training database (including training scenarios, well-known security incidents and vulnerability information, etc.), this module creates a *content description* and a *cyber range description* that define the content and activity of the training.
- *Content Definition.* This module takes the content description and generates the corresponding training content for an LMS (Learning Management System), currently using Moodle [23].

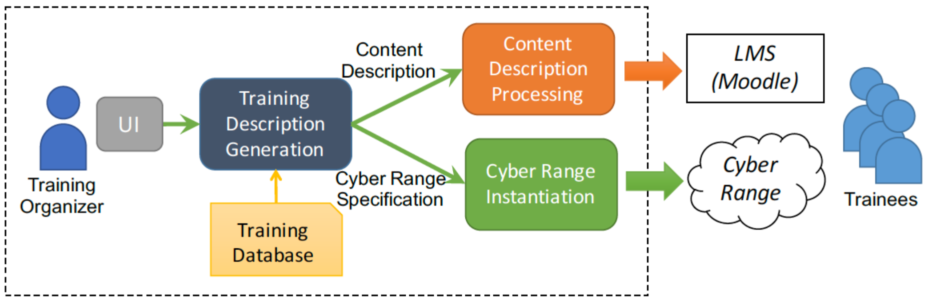

Training Organizer
UI
Training Description Generation
Training Database
Content Description
Cyber Range Specification
Content Description Processing
Cyber Range Instantiation
LMS (Moodle)
Cyber Range
Trainees

Figure 3.1: System architecture of the cybersecurity training framework CyTRONE [12].

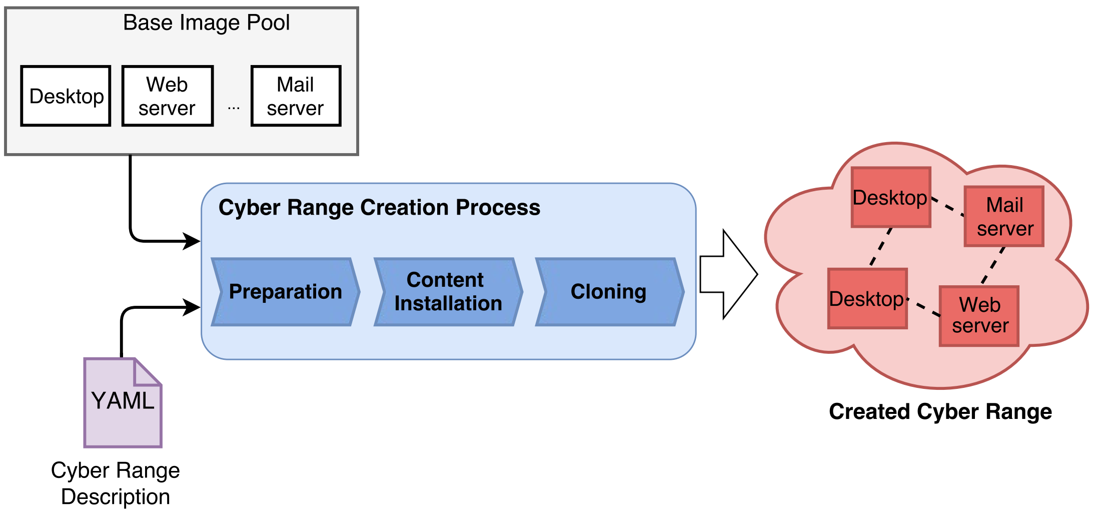


Figure 3.2: CyRIS working flow [13].

- *Cyber Range Instantiation.* This module is using CyRIS, which takes the cyber range description and automatically creates the corresponding cyber range.

The traditional approach for practical cybersecurity training is to use a dedicated and isolated physical computer infrastructure as the training environment. Such infrastructures are expensive in terms of creation and maintenance, and inefficient in terms of scalability for serving large number of trainees. Because of these significant drawbacks, using virtualization technology is a solution towards those issues, as it provides a way to create a comprehensive and realistic security training environment at a low-cost and in a scalable manner. We adapt this direction in CyRIS, as it is currently using KVM virtualization platform [25], [26] for constructing virtual cyber ranges.

3.2 CyRIS Working Flow

The working flow of CyRIS is described in Figure 3.2. The inputs, creation process and output are respectively discussed below.

3.2.1 Base Image Pool

The base image pool contains a set of virtual machine images, which are in the RAW format for KVM virtualization. They contain a pre-installed operating system and several basic system configurations (e.g., IP address, etc.). A program is created for automating the task of setting these configurations. Large companies and organizations nowadays often choose RedHat Enterprise Linux as the operating system for their servers because of the platform's performance, stability and security, which let them build their IT infrastructure across the enterprise. For this reason, CentOS 7, which is the latest community

```
- host_settings:
  - id: host_1
    mgmt_addr: 172.16.1.2
    account: crond

- guest_settings:
  - id: desktop
    basevm_host: host_1
    basevm_config_file: /home/images/desktop.xml
    basevm_type: kvm
    tasks:
    - install_package:
      - pacakge_manager: yum
        name: wireshark

    - emulate_traffic_capture_file:
      - format: pcap
        file_name: /home/trainee/traffic.pcap
        attack_type: ssh_attack
        attack_source: 2.95.120.235
        noise_level: medium

    - emulate_malware:
      - name: spyeye
        cpu_utilization: 40
        mode: dummy_calculation

    - clone_settings:
      - range_id: 123242
        hosts:
        - host_id: host_1
          instance_number: 1
          guests:
          - guest_id: desktop
            number: 1
          topology:
          - type: custom
            networks:
            - name: office
              members: desktop.eth0
```

Figure 3.3: Example of a cyber range description that provides all the necessary information for CyRIS to construct the desired cyber range.

version of RedHat Linux, is our choice as the main operating system for cyber range environments that CyRIS will prepare security-related content on.

3.2.2 Cyber Range Description

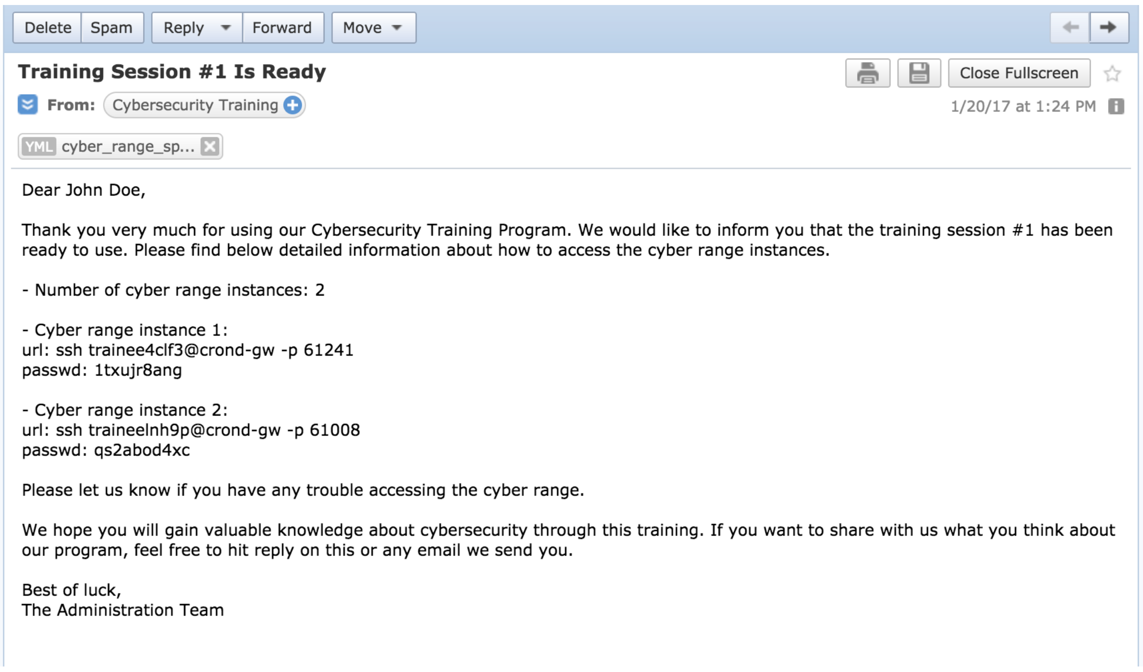
The cyber range description file is for instructors to describe the compositions and content of the cyber range. It can be created manually or by automated tools. This description is currently written in YAML, a text-file format; the reason we choose this format and not the well-known XML is because it is similar in terms of functionality, but much better in respect of readability. It defines all the necessary information needed for creating a cyber range. Figure 3.3 gives an example of how a description file looks like. It is divided into three parts as follows:

- `host_settings` contains information about the hosts that the cyber range is deployed on, including an id, a management address, and a management account.
- `guest_settings` provides information about the base images. The keyword `tasks` defines all the content of the cyber range that CyRIS needs to prepare. In this example, the cyber range consists of several settings, including installing the tool wireshark, emulating a DDoS attack, capturing traffic and deploying an emulated malware in the calculation mode.
- `clone_settings` gives details about the cloning phase. It has a unique range id, and information about hosts, guests, and network topology.

3.2.3 Cyber Range Creation Process

After having those two inputs above, CyRIS starts constructing the required cyber range. There are three main stages involved in this process, which are:

- *Preparation of the base images.* In this stage, based on the global information of `guest_settings` in the description file, CyRIS starts up the corresponding base images. It then performs some other necessary system configurations, such as copying SSH keygen, setting host name, etc., for the next step.
- *Content installation into the base images prepared above.* Here, CyRIS installs security content which is specified under the keyword `tasks` in the description file. The features that CyRIS uses in this stage will be discussed in detailed in Section 3.3.
- *Cloning of the VMs.* After finishing these two previous phases, CyRIS launches this module with the information from `clone_settings` to create a number of virtual machines and setup the network service. The details of this module will be discuss in Section 3.3.5. When the environment is constructed successfully, CyRIS sends an email notification to the instructor when the process is finished to inform them about detailed information of the environment. Figure 3.4 gives an example of how it looks like.



Training Session #1 Is Ready

From: Cybersecurity Training

1/20/17 at 1:24 PM

YML cyber_range_sp...

Dear John Doe,

Thank you very much for using our Cybersecurity Training Program. We would like to inform you that the training session #1 has been ready to use. Please find below detailed information about how to access the cyber range instances.

- Number of cyber range instances: 2

- Cyber range instance 1:
url: ssh trainee4clf3@crond-gw -p 61241
passwd: 1txujr8ang

- Cyber range instance 2:
url: ssh traineelnh9p@crond-gw -p 61008
passwd: qs2abod4xc

Please let us know if you have any trouble accessing the cyber range.

We hope you will gain valuable knowledge about cybersecurity through this training. If you want to share with us what you think about our program, feel free to hit reply on this or any email we send you.

Best of luck,
The Administration Team

Figure 3.4: Architecture of the system running CyRIS.

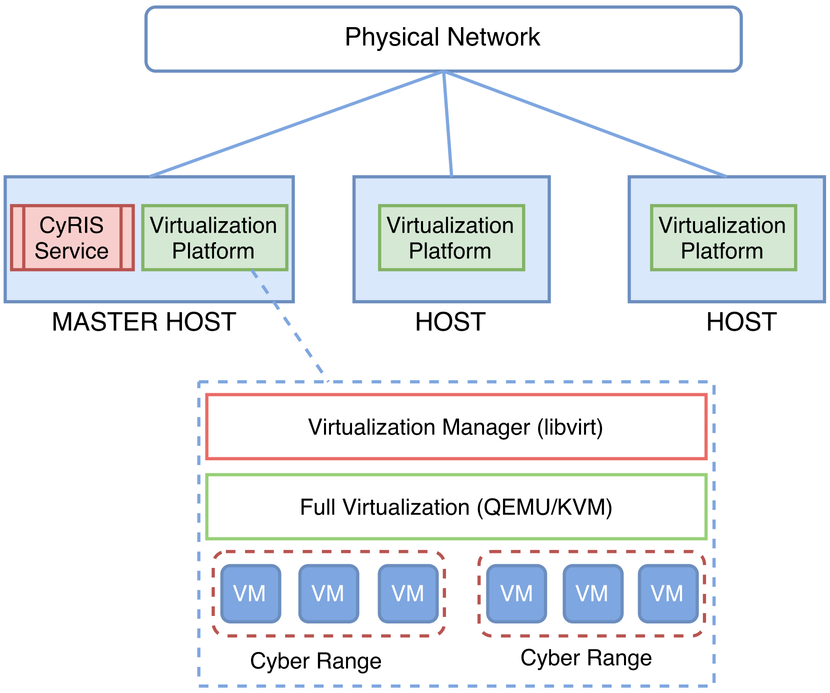


Figure 3.5: Architecture of the system running CyRIS.

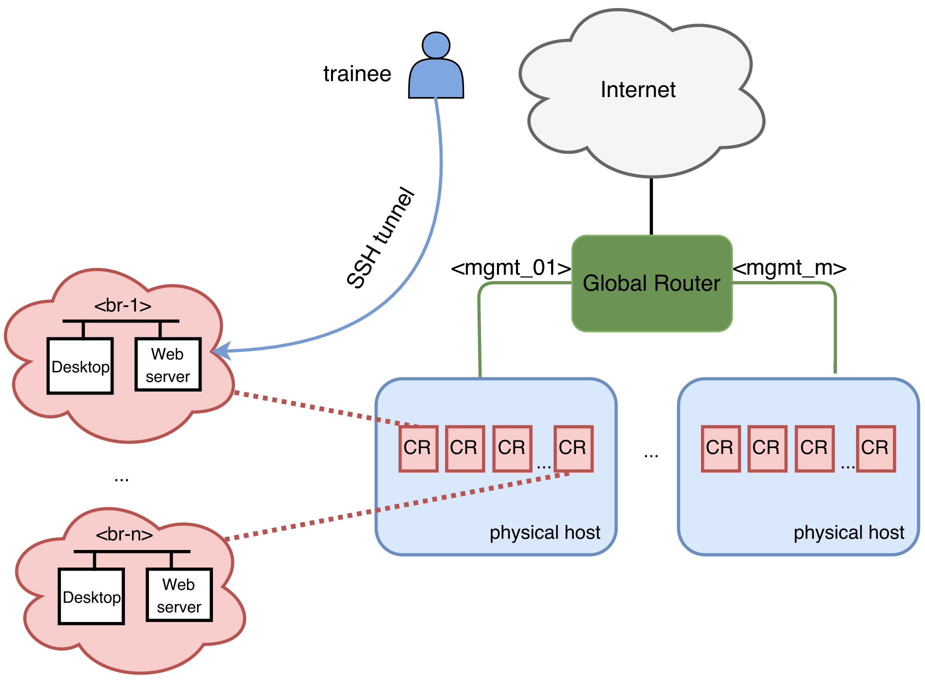


Figure 3.6: Training environment architecture. Each instance connects to its host via a virtual bridge, and trainees from outside network access their environment through SSH tunnels.

The system running CyRIS has the architecture presented in Figure 3.5. A collection of hosts, each of them is equipped with a virtualization platform (QEMU/KVM), connect to LAN network, and one is designated as a master host. This master host has CyRIS service running. This host processes the content of the description file, prepares the base images and installs security content into them, and clones virtual machines on other hosts.

3.2.4 Created Cyber Range

The architecture of a complete training environment is described in Figure 3.6. A number of cyber range instances are created, based on the requirement of the instructor, for serving multiple participants simultaneously. Each instance connects to its host via a virtual bridge, which leads to nowhere outside the host for the isolation purpose. Moreover, each instance has one SSH tunnel created during the clone phase, which connects to the outside network. Because of this, participants can easily access their environment without manually entering the host.

3.3 CyRIS Functionality

There are five main categories of functionality that CyRIS uses to install the security content into the base images and configure the network for cyber ranges: (i) system

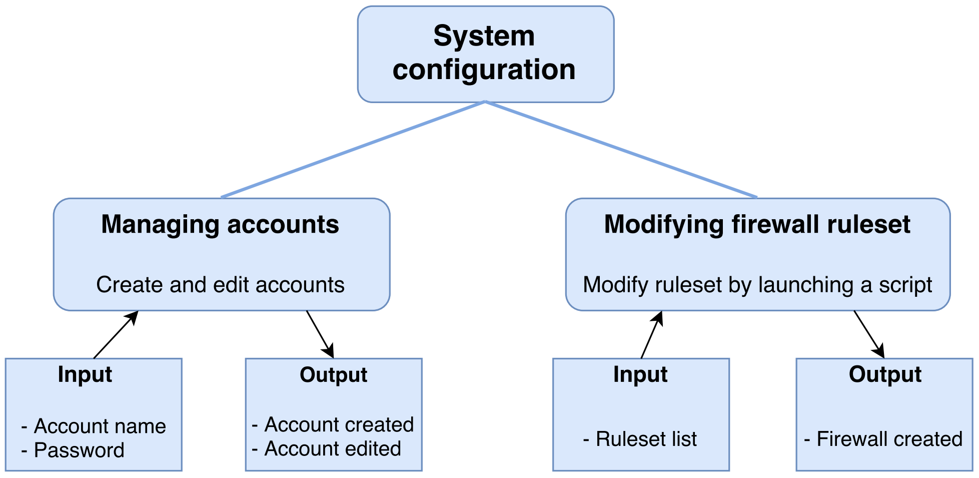


Figure 3.7: Overview of the system configuration functionality.

configuration, (ii) tool installation, (iii) incident emulation, (iv) content management and (v) clone management. Each of them has a different keyword, which is specified under `tasks` in the description file. This section describes in detail the functionality, the implementation, and the usage of each of them.

3.3.1 System Configuration

As being said in Section 3.2.1, basic system configuration (host name, ssh keys, ip address, etc.) in base images has been taken care in advance. Thus, CyRIS tasks about system configuration, which are about creating settings for security training, are limited to managing accounts and modifying firewall rules (Figure 3.7).

Managing accounts is used to create new accounts and edit information of existing accounts. These settings can be used for practicing penetration testing techniques, when trainees identify potential weak passwords in the system and learn how to crack them.

Another function in this category is modifying firewall ruleset. In practice, firewalls provide a critical layer of security that protects a system against threats, in which it filters the incoming and outgoing traffic, controls the list of opening ports, and keeps a close watch on running softwares and tools. In cybersecurity training, it is essential to teach participants the importance of firewall systems, as well as common security mistakes that people often make when configuring the ruleset.

One way for instructors to prepare hands-on activity exercises for this topic is to have a firewall set up in the cyber range, which has a mixture of good and bad rules. Trainees then can move on to understanding the meaning of each rule, and trying to spot if there is any potential security hole in this setup. In this perspective, CyRIS can help the instructors by providing the function of setting up and modifying firewall ruleset. The implementation at the current stage is using `iptables` [29] software, which is pre-installed in Linux operating systems. The instructors simply specify an absolute path to a script that contains a list of rules. This provides them with an automated way for adding and

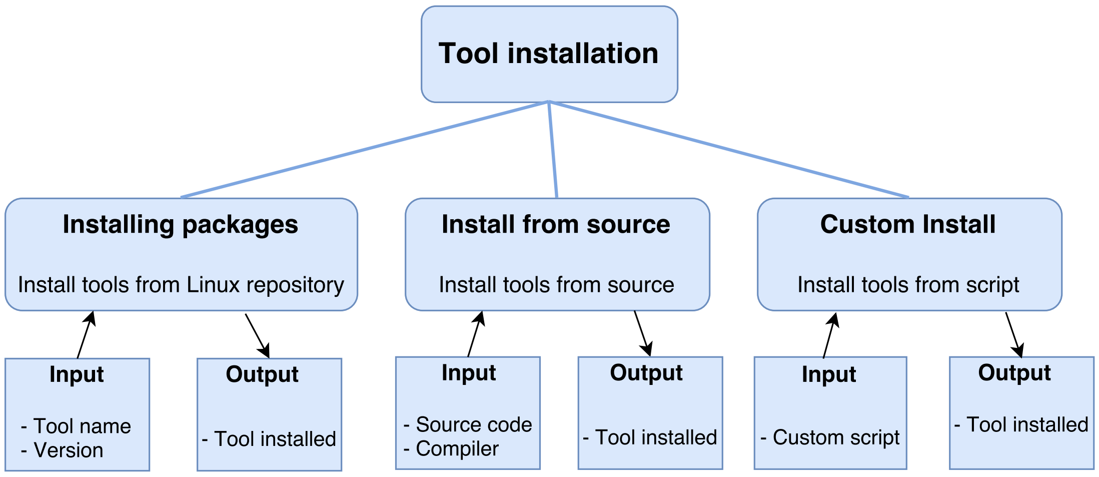


Figure 3.8: Tool installation functionality, which offers three ways of installing a package: from official repository, from source, and from custom script.

modifying firewall rules, so that they can have a realistic training environment in a flexible manner.

One typical question that might be raised in this context is why the instructors are required to write down a list of rules to a script in advance, and not be able to specify them directly in the description file. Note that in many cases, it requires a number of firewall rules to protect a complex infrastructure. Giving them directly in the description file might make it long and difficult to read afterwards. Therefore, putting firewall rules into a separate script is our option, as it becomes convenient for the instructors to manage their rules, for example when they want to separate good and bad rules for some purpose.

3.3.2 Tool Installation

In cybersecurity, tools are essential. Various tools are well-known and indispensable in the security world, such as `wireshark` and `tcpdump` for network sniffing related jobs, or `aircrack-ng` suite and `john-the-ripper` for penetration testing. Knowing how to use them is a must for people working in this field, and therefore it is one of the main topics of training sessions.

This function provides the instructors with a mechanism to install such tools automatically. As described in Figure 3.8, this function contains three main types of installation manner, which are:

- *Package installation*, in which CyRIS installs tools from the official Linux repositories using package managements (apt-get, yum, etc.). Instructors specify the type of package management, the name of software and its version, then the module automatically downloads and installs it on the machine.

- *Source installation*, which builds tools directly from source. Instructors provide an absolute path that leads to the source and a version of the compiler, then the system

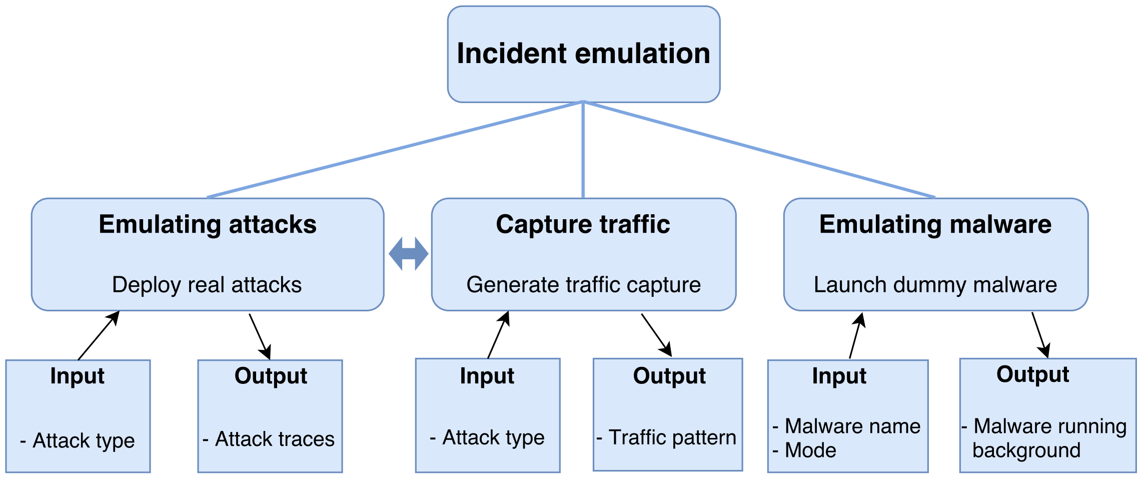


Figure 3.9: Sub-functions of the incident emulation category.

installs it by `make install` commands. However, before using this function, the instructors need to make sure that all dependencies needed are probably installed in advance, as CyRIS has no function to solve those automatically. Therefore, it is recommended for instructors to try to install the tool on a local machine first, so that they know a list of dependencies. After that, they can ask CyRIS to install those using the previous function, and everything will be set for their custom tool.

- *Custom installation*, which gives instructors freedom to install any tools in any ways they want. They basically give a script containing all of the necessary commands, and CyRIS then executes and builds tools from there. The details of this function's implementation will be discussed in Section 3.3.4

In the cybersecurity world, there are many tools that are customized, and it is difficult to install them via package managements. Moreover, in many cases that instructors want to have an older version of a tool that contains a vulnerability, and it no longer exists in official repositories due to the security policy. Therefore, having the two latter installation methods helps the instructors facilitate the task of installing tools in special situations.

3.3.3 Incident Emulation

This group includes three main functions, which are (i) attack, (ii) traffic capture and (iii) malware emulation (Figure 3.9). They have the ability of launching actual incidents to prepare security-related content for cyber ranges, and they stand as the main differences between CyRIS and other well-known automated environment configuration tools. This section describes in detail these functions' designs and implementations.

3.3.3.1 Attack Emulation

Recognizing attack patterns is one of the main activities during cybersecurity training. It helps trainees improve the ability of detecting whether any attack is taking place, and of defending the system against attacks. Two kinds of attack that are usually deployed in a training environment: static and dynamic attacks.

Static attacks are actions that take place before the training session starts to produce logs, traces, or files that correspond to the attacks. These evidences, throughout the actions, are recorded and left in the system, so that trainees can try to investigate and find out what kind of attacks has been performed. The purpose of this activity is to improve trainee 's knowledge in forensic techniques (log and ruleset review, network sniffing and file integrity checking), as well as to help them understand patterns of different types of attack. On the other hand, dynamic attacks are usually deployed live during the training session to enhance trainees' instant response. For example, white-hat experts perform a DoS attack in order to reduce the performance of the system, and trainees have to know how to recognize they are being under attack and how to defend against it.

CyRIS is capable of emulating static attacks of specific types, including:

- *SSH dictionary.* A python package, which is called `paramiko` [30], is being used in the current implementation. To trigger this kind of attack, the instructors specify an account name as a victim and a number of attempts. CyRIS, after having those inputs, creates a bunch of threads, starts attacking the victim, and automatically generates logs as consequences.

- *DoS and DDos.* A tool called `hping3` [31] is involved in the current implementation. Similarly to the previous case, a victim needs to be given for performing the attack.

Besides, CyRIS offers the instructors an option for capturing traces of these attacks, which can prepare settings for related questions. This will be discussed in the next section 3.3.3.2.

We decided to start CyRIS with emulating these two kinds of attack is because they are among the most popular network attacks in 2016, as the first type took 19 percent the total number of attacks having taken place, while the second were 16 percent [10]. Therefore, they are often a topic in cybersecurity training, which is to teach participants the attack patterns and the methods to deal with them. Moreover, they requires simple tools (`paramiko` and `hping3`) for emulating. Regarding other types, such as browser, SSL, back door attacks, etc., as they are more complex, CyRIS has no function to emulate them in the current stage. It however could help the instructors create vulnerability corresponding to those attacks in the cyber range by deploying the content management functionality (Section 3.3.4), and participants are able to practice with the vulnerability by launching an attack by themselves, or defending against white-hat hackers or their opponents during the training.

We plan to extend this function to be able to emulated other types of attack, as well as to perform them dynamically. Regarding wireless connection, because of having no way to mimic a wireless network among virtual machines in KVM platform, emulating attacks in this kind of connection is out of CyRIS' reach at the current stage.

3.3.3.2 Traffic Capture

Alongside with attack emulation is the traffic capture function. With this, CyRIS is able to prepare traces for various kinds of attack in both wired and wireless connection networks.

For wired connection, users define what kind of attack traces they want in the cyber range (SSH dictionary, DoS, DDoS) and CyRIS deploys the corresponding attack emulation. Simultaneously, it uses `tcpdump` to sniff the network traffic and capture traces of the attack under the `pcap` format. These traces can be given to trainees so that they can practice their forensic skills by trying to find out the pattern of the attack. Moreover, CyRIS offers instructors an option of mixing the attack traces with usual network traffic, making the capture file more realistic, and also making the problem more challenging to participants.

Regarding wireless connection, because of the problem that has been said in the previous paragraph, we prepare in advance traffic capture files that contain traces of attacks that are normally performed in real world, which are *replay attack* and *DoS attack*. In addition, we prepare files related to WEP wireless network security protocol, so that trainees can investigate and learn how to crack passwords using tools like `aircrack-ng`. We also produce files of other types of protocol (WPA, WPA2) for reference purposes.

3.3.3.3 Malware Emulation

Detecting malicious programs is another important technique for cybersecurity professionals. It is essential to know the methodologies to discover whether any unusual application is running under the hood.

With the malware emulation function, CyRIS offers the ability of launching an emulated malware running in the system. This dummy malware is totally unharmful and can run under two modes at the current stage: performing a calculation or listening to a port. In the first mode, the dummy malware is able to run with a certain amount of CPU consuming, neither too low to avoid being invisible from trainees, nor too high to appear obvious. The second mode allows the malware to run as a service that listens to the network traffic via an arbitrary port. To deploy this dummy malware, instructors give it a name and a mode. It then appears in the list of background processes, creating a security threat in the system.

Besides, instead of using the dummy malware, there are cases when instructors want to deploy their custom malware for training participants about particular security topics. CyRIS helps them satisfy this need by having the content management group of functions (which will be discussed in the Section 3.3.4), in that instructors simply use an executing scripts mechanism to launch their custom malicious programs in the cyber range.

3.3.4 Content Management

This category offers three mechanisms to modify the content of an environment, including (i) copying content and (ii) executing scripts (Figure 3.10).

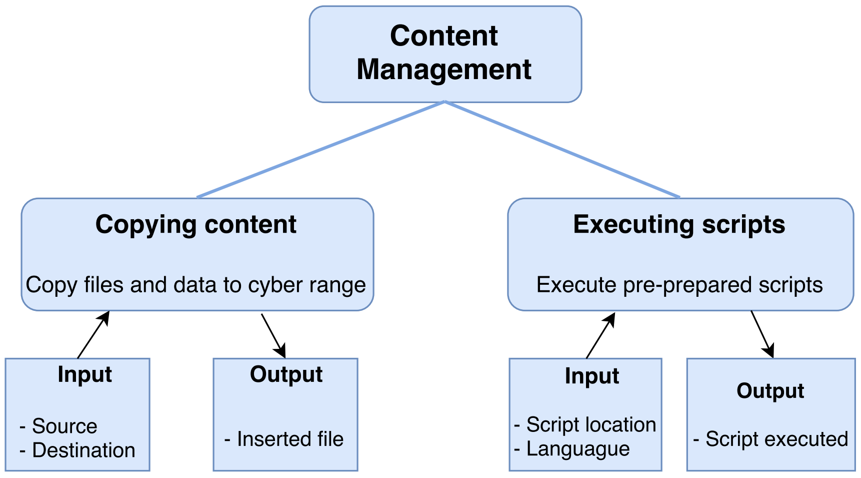


Figure 3.10: Sub-functions of the incident emulation category.

Basically an OS is a collection of files in a file system. In other words, it is possible to have every setting by inserting the correct files into the correct place [21]. CyRIS adopts this method by having the copy content function. It allows instructors to copy files and data from outside to the cyber range to create a setting. For example, if they want to set the host name of a machine, they simply use this function to copy a text file named `hosts` with the host name inside and copy it to the `/etc/` directory.

The second mechanism is the ability of letting instructors execute scripts in cyber ranges. It needs two parameters, which are `program` and `compiler`. The `program` tells CyRIS the location of the script and arguments needed, while the latter one specifies the script language, including shell script, Perl, Python and Ruby.

3.3.5 Clone Management

After finishing preparing settings for cybersecurity training on base images, the clone management functionality takes place to clone a set of virtual machines from the above base images, and set up other configurations to construct a complete cyber range. This section describes in detail the design and implementation of this function.

For a cybersecurity training organization, it is common to have multiple training programs running in parallel. Each program is about a different topic, and therefore the security content needed in the cyber ranges also vary from one to the other. Hence, it is important to have a unique name for each of them for the ease of managing purpose. In CyRIS, each cyber range is assigned a unique ID number by the instructors. This ID, beside the purpose of differentiating among cyber ranges, is also used for naming related configurations of that cyber range, such as names of virtual machines, virtual bridges, etc. Moreover, the IP addresses of the virtual machines, during the cloning phase, will be set automatically from the ID, based on a list of pre-defined rules in CyRIS. Two purposes of this design are: (i) to reduce the work of the instructors that they need not to manually set IP addresses for each virtual machine in the cyber range, and (ii) avoid as much as possible human mistake, as the instructors can accidentally specify the same IP address for multiple virtual machines.

In cybersecurity training, there are many topics related to network security which correspond to many kinds of network topology. In this function, CyRIS provides instructors the ability to create various network topologies, from the simple ones such as bus, ring, star, to the complex ones as DMZ topology. Moreover, in complicated topologies , it often requires to have a gateway or a firewall in the system, in which it is assigned the task of filtering and forwarding traffic among different network segments. CyRIS satisfies this need by allowing the instructors to specify a set of necessary forwarding rules to any machines in the environment. The current implementation, however, only allows simple rules, which contains basic parameters such as the source and destination machines, and the source and destination ports.

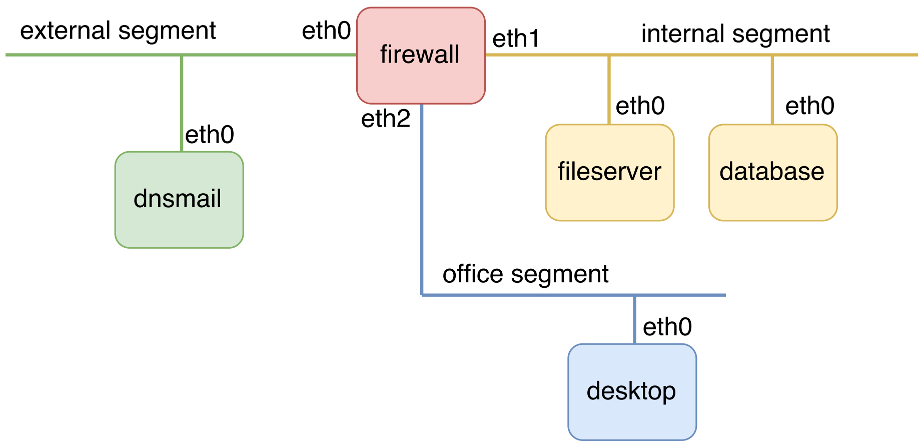
The working flow of cloning a cyber range is as below:

1. *Cyber range directory creation.* Based on the ID of the cyber range, CyRIS creates a directory in all the physical hosts. The name of the directory is made unique by associating with the ID. After that, all of data which relates to the cyber range will be stored in its corresponding directory.

2. *Base image copy.* After the first two stages finish in which all the security content have been installed on the base images, CyRIS starts copying them from the master host to others into the corresponding directory, using `parallel-scp` [32] commands for parallelizing the process. Note that if the cyber range contains more than one base image, then they are also copied in parallel.

3. *Parallel clone execution.* CyRIS first in parallel creates necessary scripts for the cloning process in the hosts, using `parallel-ssh` [32]. These scripts are used for tasks of creating virtual bridges to connect the cyber range to its hosts, cloning virtual machines from the base images, generating SSH tunnels from the outside network to the cyber range, and configuring forwarding rules on any machine if they are specified. CyRIS then executes these scripts in a pre-defined order, and also in parallel on multiple hosts, to construct the complete cyber range.

An example of an DMZ network topology clone is described in Figure 3.11. Five types of virtual machines are involved in this scenario, which are divided into three network segments: an dnsmail server in external segment, a file server and a database server in internal segment, and a desktop in office segment. The gateway server plays the role of a network monitor, in which it helps segments communicate with each other by forwarding traffic among them based on pre-defined rules. These rules are listed under the keyword `forwarding_rules`. For example, servers from the external segment only able to send traffic to the database server and the file server via port 3306 and 139 respectively, etc.



(a)

```
- clone_settings:
  - range_id: 112
    hosts:
    - host_id: host_1
      instance_number: 1
      guests:
      - guest_id: firewall
        number: 1
        forwarding_rules:
        - rule: src=office,external dst=internal.database dport=3306
        - rule: src=office,external dst=internal.fileserver dport=139
        - rule: src=office dst=external_servers dport=25,53
      - guest_id: dnsmail
        number: 1
      - guest_id: fileserver
        number: 1
      - guest_id: database
        number: 1
      - guest_id: desktop
        number: 1
      topology:
      - type: custom
        networks:
        - name: external
          members: dnsmail.eth0
          gateway: firewall.eth0
        - name: internal
          members: fileserver.eth0, database.eth0
          gateway: firewall.eth1
        - name: office
          members: desktop.eth0
          gateway: firewall.eth2
```

(b)

Figure 3.11: DMZ network topology and the corresponding cyber range description.

Table 3.1: Current functionality of CyRIS

Categories	**Basic functions**	**Security functions**
System Configuration	Manage accounts	Modify firewall ruleset
Tool Installation	Install package Install from source Custom install	
Incident Emulation		Emulate attacks Capture traffic Emulate malware
Content Management	Copy content Execute script	
Clone Management	Configure network Clone virtual machines	

3.4 Discussion

We summarize all CyRIS functions in Table 3.1, in which they are divided into two groups that are basic and security ones. The first group contains common functions for environments configuration tasks, while the second group are to use from a security perspective. These security functions are the main differences in CyRIS compared to other well-known tools. In conclusion, there are several requirements that CyRIS meets for constructing a realistic cyber range:

- *Content installation.* By offering a set of functions, which covers both basic operations (installing tools, copy data, etc.) to advanced ones (emulating attacks, capturing network traffic, etc.), CyRIS greatly facilitates the task of preparing realistic content for a cybersecurity training program.
- *Network topology.* A cyber range consists of a set of connected virtual machines that mimics a real network environment, and its topology can be various. In the current implementation, CyRIS allows instructors to specify many types of topology, in an easy and convenient manner. Moreover, CyRIS offers an option to configure forwarding traffic rules on any machine in the environment, which is often the need in complicated network topology.
- *Environment separation.* Cyber ranges are places for trainees to practice all kinds of security techniques, and it is possible to have traffic leakage to the outside network. To avoid this problem, it is important to isolate the training environment. In CyRIS, cyber range instances connect to the host through virtual bridges that lead to nowhere outside the host. Moreover, to improve the fairness during the

training, these virtual bridges have no connection between each other, and an account and a password are generated randomly for each trainee to access their cyber range instance via SSH connection, making sure that no one is able to access others' environments.

- *Parallel execution.* For large cybersecurity training program which involves hundreds participants, it is required to create the corresponding number of cyber ranges in a reasonable amount of time. For achieving this requirement, CyRIS provides the ability of cloning virtual machines on multiple hosts in parallel, using the tool called `parallel-scp` [32]. The details about its efficiency is discussed in Chapter 4.

- *Informative notification.* As being shown in Figure 3.4, after the creation process is finished successfully, an email is sent to the instructors for informing them the information about the total number of instances that they have created, alongside the details how to access each of them.

We provide a full sample of the cyber range description file in Appendix A, in which it contains all functions that CyRIS offers at the current stage, along with their usage and keywords. Please refer to that for more information.

Chapter 4

System Evaluation

In this section, we first evaluate the coverage that CyRIS is able to offer in terms of preparing security content for cybersecurity training. For this purpose we use the U.S. NIST Technical Guide to Information Security and Testing Assessment [24] as a reference. We then discuss about the feature comparison between CyRIS and other tools. In addition, we present results of CyRIS performance in creating representative cyber ranges.

4.1 Functionality Evaluation

This section describes our evaluation of CyRIS about feature coverage in preparing content for cybersecurity training, and the comparison between CyRIS and other similar tools in respect of functionality.

4.1.1 Feature Coverage

The NIST guideline [24] states a number of techniques in information security testing and assessment, which are categorized into three main groups:

- *Review techniques* relate to manual inspections and reviews to evaluate applications, architecture designs of network and systems in the purpose of discovering vulnerabilities. This group of techniques consists of documentation, log, ruleset, and system configuration review; network sniffing; and file integrity checking.

- *Target identification and analysis techniques* are testing techniques that can identify systems, ports, services, and potential vulnerabilities, and may be performed either manually or using automated tools. They include network discovery, network port and service identification, vulnerability scanning, wireless canning, and application security examination.

- *Target vulnerability validation techniques* are testing techniques that corroborate the existence of vulnerabilities, and may be performed manually or by using automatic tools, depending on the specific technique used and the skill of the test team. Target

Table 4.1: Summary of CyRIS functions required to support each security technique in the U.S. NIST Technical Guide to Information Security and Testing Assessment

NIST Security Technique	Basic Functions					Security Functions				
	Manage Accounts	Install Tools	Copy Files	Execute Scripts	Configure Network	Generate Logs	Modify Firewall	Emulate Malware	Emulate Attacks	Capture Traffic
Log Review		x		x		x			x	
Ruleset Review		x		x			x			
System Configuration Review	x	x		x		x	x			
Network Sniffing		x	x		x				x	x
File Integrity Checking		x	x	x						
Network Discovery		x			x					x
Network Port and Service Identification				x	x		x	x		
Vulnerability Scanning		x	x		x	x	x	x	x	x
Wireless Scanning		x	x						x	x
Password Cracking	x	x								
Penetration Testing	x	x	x	x	x	x	x	x		
Social Engineering	x		x							

Table 4.2: Functionality comparison between CyRIS and other similar tools

Tools	**Content installation**		**Network setup**	
	Basic functions	**Security functions**	**Physical host**	**Virtual machine**
Ansible, Chef, Vagrant	✔		✔	
Openstack, Spring OS			✔	
Alfons	✔		✔	✔
CyRIS	✔	✔		✔

vulnerability validation techniques include password cracking, penetration testing, social engineering, and application security testing.

Table 4.1 shows in detail about how combined CyRIS functions are used to create content for different security techniques. Basically, all realistic content needed for each security technique in the NIST guideline is covered by CyRIS. Basic functions like install tools and so on play the role of preparing the infrastructure for the system, and the security group prepares specific content that corresponds to each and every technique.

One example is training regarding the network sniffing technique. Normally to train for mastering this technique, a traffic capture file with some attack pattern is given to trainees. CyRIS first creates the required file by combining the attack emulation and traffic capture functions. It then provides a way for trainees to investigate the file by installing `tcpdump` or `wireshark`, depending on the specification of the instructors. Another example is about vulnerability scanning technique, in which trainees learn how to identify vulnerabilities in the system (e.g., malware applications, open ports, etc.). In this case, CyRIS either executes a script to start an application or deploys the dummy malware that has an unusual name and listens to an arbitrary port.

4.1.2 Feature Comparison

Table 4.2 shows a comparison in terms of functions between CyRIS and other recent similar tools. We divided them into two categories, which are content installation and network setup functionality. The first group reflects the ability of performing operations to creating content in individual nodes in order to create a desired environment. These operations, however, exclude the function of installing an OS in a node from scratch, as it is a pre-prepared step in advance. The network setup functionality includes two main types of functions, which are physical and virtual configuration. While the first one refers to tasks that relates to configure network service on a physical node, such as setting up physical interfaces, VLANs, etc., the second type mentions about tasks on a virtual machine, such as setting up its interfaces, constructing virtual bridges, and so on.

Content Installation
Regarding this category, while basic functions are common in automated environment configuration tools, we find that security functions are unique to CyRIS. There are many security settings Alfons can prepare by executing pre-prepared scripts or copying data files from outside to the cyber range, but this process is costly and requires instructors to generate such files manually in advance. With the security functions, CyRIS allows them to create these settings by launching real actions, in a convenient and flexible manner. This characteristic is extremely important and helpful in creating a realistic environment for cybersecurity training.

One example of its usefulness is preparing logs for unsuccessfully login attempts in CentOS 7. Alfons or Ansbile can simulate this situation by replacing the file `/var/log/secure` with another one containing logs for this incident that instructors have created in advance. This method is inefficient, for two main reasons. Firstly, it requires instructors to perform such an attack and generate logs by themselves. Since they have many courses running one after another, the timestamp in the logs they have created, at a certain time, will run out of date. As the result, they have to do the procedures again and again, which is a tedious and error-prone task. Secondly, this method works only with people who have the habit of investigating the file `/var/log/secure` in the system. In CentOS 7, one may use a command called `lastb` to check for such attempts, and none of this trace is shown at the output. In contrast, by emulating the ssh attack, CyRIS automatically generates real logs for the incident and the information appears in both places.

Another example is creating necessary content for training participants about network sniffing technique. Again, Alfons is able to do this by having a pre-prepared traffic capture file that contains an attack pattern, such as a DDoS attack, and then copying it into the cyber range. However, the same problem appears as before, that this approach requires instructors to, somehow and by some tools, host a DDoS attack and capture the traffic. This preparation is complicated and dangerous, especially with people who have little experience, as it might cause serious problems if the attack is not controlled carefully. However, CyRIS, with the built-in functions for emulating attacks including DDoS type, lets instructors to produce such files automatically in an easy manner. Moreover, CyRIS provides some additional modules, that allow to combine this specific captured attack pattern with normal traffic. By doing this, the attack pattern will be more difficult to detect, as it will make the problem more challenging for the participants.

Network Setup
In the network setup functionality, CyRIS is able to create virtual machines in cyber range environment and configure network among them. The network topology module is able to mimic wide range of topology, such as bus, ring, or DMZ. Moreover, CyRIS makes the environment isolated from the outside network to avoid potential traffic leakage or bad incidents. It then can generate a random account and password for each trainee so that they cannot access others' information, enhancing the security and fairness during the training.

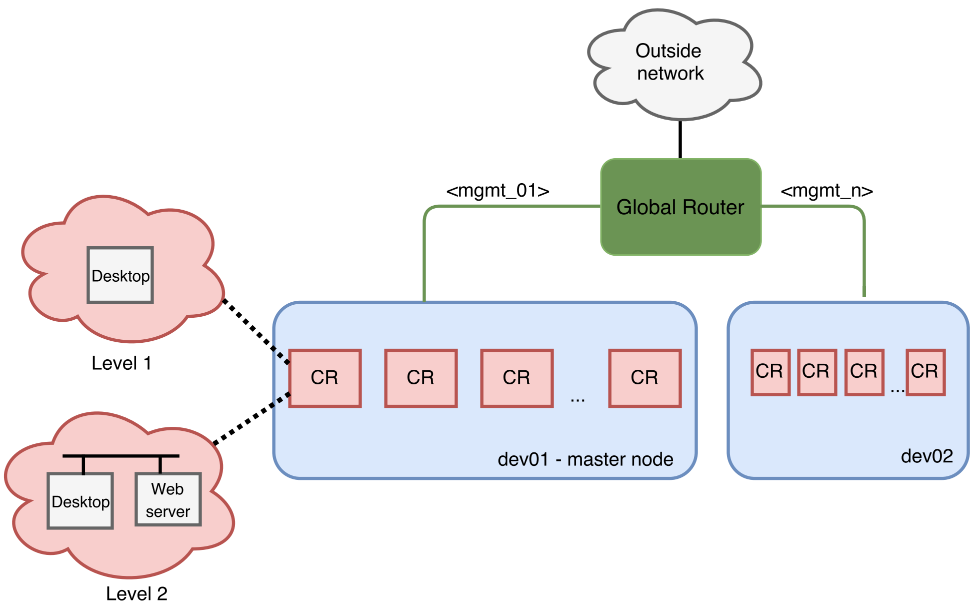


Figure 4.1: Network topology for performance evaluation.

4.2 Performance Evaluation

This section describes our evaluation in respect of CyRIS performance. These experiments are conducted in StarBED testbed of National Institute of Information and Communication Technology [20]. Our evaluation is divided into two parts, which are:

- *Large-scale performance evaluation.* We firstly use CyRIS to construct several representative cyber ranges for large-scale scenario, in order to measure the execution time for each phase. By this, we are able to analyze CyRIS performance ability in detail.
- *Comparison to Alfons.* We, after the first experiments, use CyRIS to construct a cyber range, which is similar to the environment Alfons used for its system evaluation [21]. We then compare the creation time between the two, and give our analysis.

4.2.1 Large-scale Performance Evaluation

4.2.1.1 Experimental setup

We construct an environment as shown in Figure 4.1 for the evaluation. The physical facility consists of 30 servers (from `dev01` to `dev30`), connected to a global router. Each server has the specifications of two 4-core Intel Xeon®E5504 2GHz CPUs, 72GB memory, 400GB HDD hard drive, and 1Gbps network interface. The first server `dev01` is designated as the master host for running CyRIS.

As having mentioned before in CyRIS working flow, the cyber range instantiation process is divided into three main stages:

- Preparation of the base images for VMs, which is conducted in one host, the master host `dev01`;
- Content installation into the base images above, also conducted on the master host;
- Cloning of the VMs on multiple hosts, which includes the tasks of copying base images from the master host to others, cloning VMs from those base images, configuring the network service, and starting up the VMs.

Cyber Range Organization and Design (CROND), an NEC Corporation endowed chair at JAIST, has been developing a cybersecurity training for information security professionals. Its content has multiple difficulty levels, covering all the essential security techniques mentioned in the NIST guideline. Based on this training, we consider two typical levels for our experiments. The first level's topic is about the security of a desktop computer. Its cyber range includes one desktop, and contains content for review and analysis techniques training, such as log and system configuration review, network sniffing, vulnerability scanning, etc. The second level is designed towards advanced security people, in that it concentrates on more sophisticated security knowledge about networking. Its cyber range reflects a small company's network, which has one web server and one desktop connected to each other, and contains content for skills including network discovery, password cracking, penetration tesing, and social engineering techniques.

We use these two models of cyber range in the first part of our performance evaluation. Two aspects of CyRIS we would like to study, which correspond to two scenarios, are described as below:

- *Effects of parallel execution.* As being described in the cloning function implementation, we use several techniques for parallelizing the execution of CyRIS on multiple hosts. To evaluate its efficiency, we conduct experiments with a fixed number of virtual machines (20) that are instantiated on 1, 2, 5, and 10 hosts (with 20, 10, 4, and 2 VMs per host, respectively).
- *Effects of large-scale execution.* In this series of measurements, we keep the number of VMs per host constant (20), and assess performance for a large-scale scenario with up to total of 600 VMs on 30 hosts (representing 600 cyber ranges for Level 1, or 300 cyber ranges for Level 2).

4.2.1.2 Experimental Results

We first discuss the effect of distributed execution in CyRIS, and then analyze its performance in constructing large-scale cyber range environments.

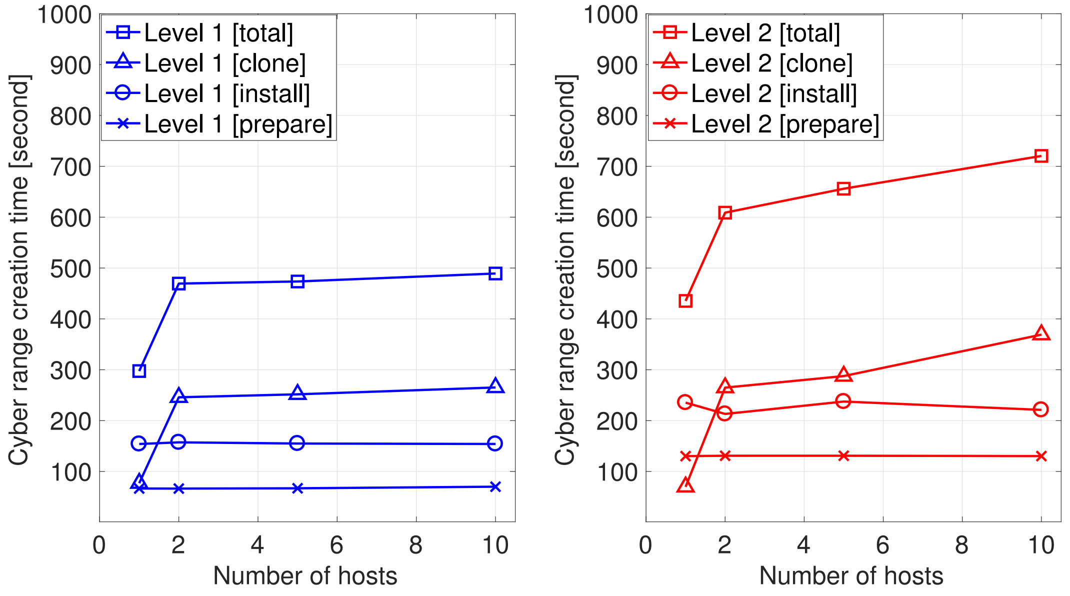


Figure 4.2: Cyber range creation times in Level 1 and Level 2 for studying the effects of distributed execution, which a fixed number of virtual machines (20) are instantiated on 1, 2, 5, and 10 hosts.

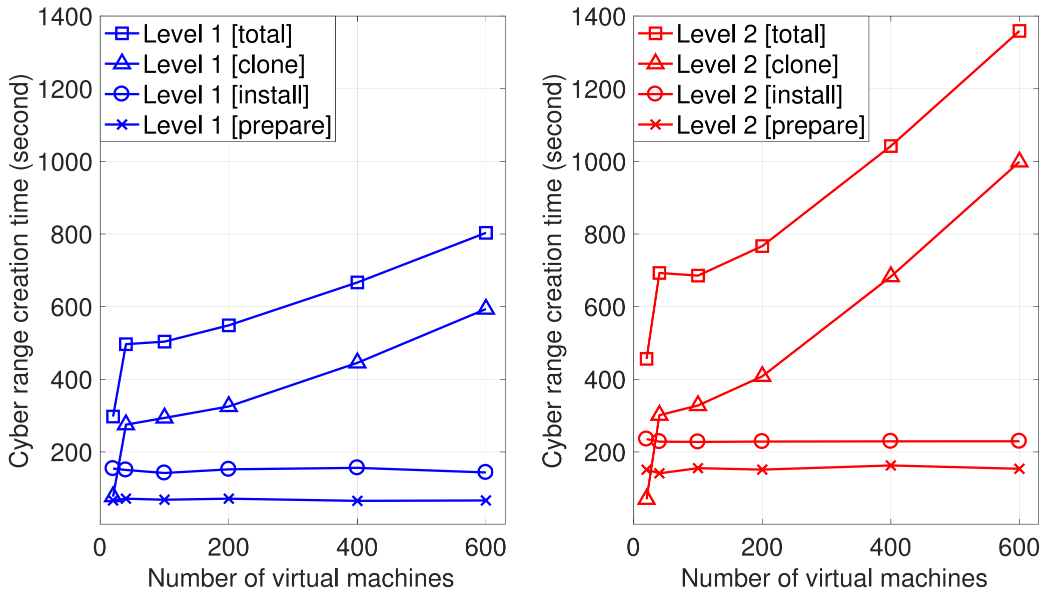


Figure 4.3: Cyber range creation times in Level 1 and Level 2 for testing CyRIS performance under large-scale creation scenarios. The number of virtual machines per host is constant (20), and the total number is up to 600 VMs on 30 hosts.

Effects of parallel execution
Figure 4.2 shows the average time required to instantiate the cyber ranges in the first series of experiments, for each step (preparation, content installation, clone) and in total. It is understandable that the preparation time for Level 1 is nearly as half as for Level 2, since there are one virtual machine is needed in the former whereas two are used in the latter. The content installation phase, similarly, observes the same trend when the time spent for Level 2 is higher, even though not double as the previous case, than for Level 1. This result, in fact, is expected because the content necessary for Level 2 is more sophisticated and complex than for the other.

Regarding the result of the cloning phase, it is interesting to observe that the spent time for Level 1 and Level 2 are quite similar, except for the right-most case. As having described in Section 3.3.5, two main tasks, which are copying base images from the master host to others and cloning virtual machines from the base images, are included in this phase. Note that the task of copying the base images is done in parallel among hosts, and if there are more than one base image, then their copy processes are also done in parallel. The same appears in the latter task, as the process of cloning virtual machines on multiple hosts, and among multiple virtual machines on one hosts, are proceeded in parallel.

In the first experiment when only one host, which is the master host itself, is involved in the scenario, there is no need to copy the base images. Thus, the times needed for this case in both levels are rather low, with roughly 1 minutes. When it comes to more hosts, even done in parallel, the copy process starts showing its effect. In case of 2 and 5 hosts, the clone time in Level 2 are about 4.5 and 4.9 minutes respectively, which are slightly longer than in Level 1 with roughly 4 minutes in both the cases. The most significant effect is observed in the case involving 10 hosts, which the clone time in Level 2 (6 minutes) is considerably higher than in Level 1 (4.5 minutes). These are expected results due to the fact that two images are required to copy simultaneously in Level 2, which leads to the problem of decreasing the throughput because the network bandwidth and NIC (network interface card) capability on the master host are overloaded.

The total construction time, following the results that are discussed above, yells no surprise in both Level 1 and Level 2. Note, however, that after the cloning phase, there are several tasks CyRIS has to do in order to construct a complete cyber range: creating SSH tunnels from the network outside to the cyber ranges, generating random passwords for each instance, and sending out the detailed information to the instructor. The total time, therefore, is slightly larger than the sum of the three previous phases. In most cases the total construction process finishes in a reasonable time of under 10 minutes, and only in the worst case (Level 2, 10 hosts) the time reaches about 12 minutes.

Effects of large-scale execution
Figure 4.3 shows the results from the second series of measurements. Regarding the preparation and content installation phases, basically no difference is presented compared to the lower-scale experiments shown before, since the operations are similar in both the cases.

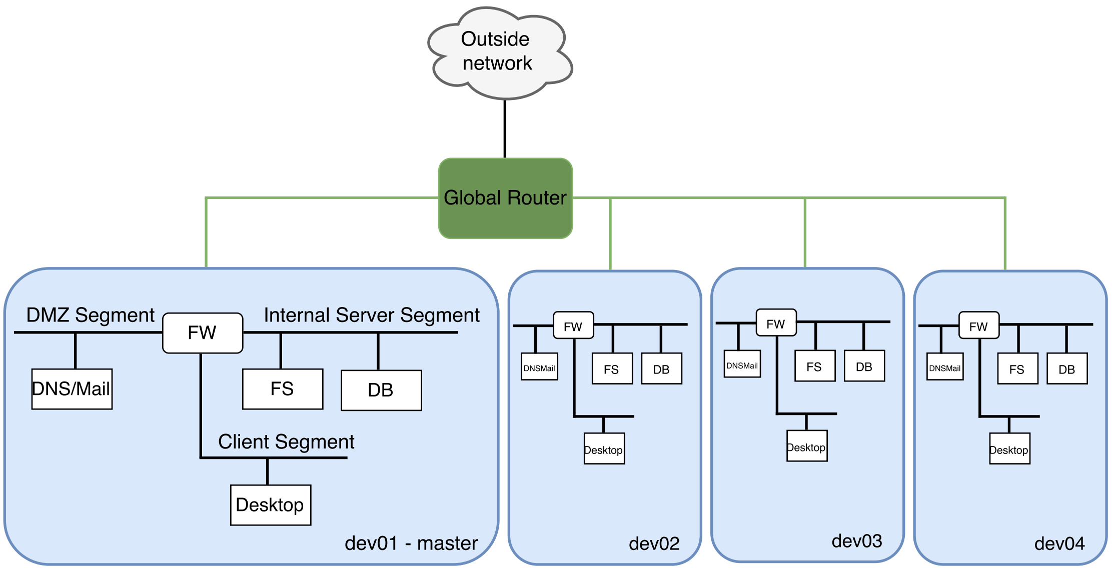


Figure 4.4: DMZ network topology used in experiments for comparing CyRIS and Alfons.

The cloning phase, however, yells an exponential increase for the required time, with a higher exponent for Level 2, which requires copying a double number of the base images. This is, again, explained by the similar problem appeared before: the throughput decreased when too much amount of data are transmitted across the network at the same time, as the network bandwidth and NIC capability are overloaded.

Nevertheless, the total time results show that, in a setup for 100 participants, the construction can be finished in under 10 minutes for Level 1 (100 VMs), and in under 15 minutes for Level 2 (200 VMs), which are reasonable durations in our opinion given the size of the program is relatively large. In the extreme case of using 600 VMs, for Level 1 (600 trainees) the setup is completed in under 15 minutes, and even for Level 2 (300 trainees) the setup is completed in about 22 minutes.

4.2.2 Performance Comparison

As having said in Chapter 2, Alfons is the closest tool to CyRIS, as it is equipped with functions for the same purpose of facilitating the cyber range preparation task for cybersecurity training. We have compared these two tools in terms of functionality in Section 4.1.1, and this comparison is conducted in respect of performance.

4.2.2.1 Experimental Setup

The environment for this evaluation is shown in Figure 4.4. Four physical machines are involved in this situation, and they have the same specs as described before. Each physical machine contains a training environment, which reflects a small DMZ topology with three network segments, namely the client, internal server, and DMZ. There are

Table 4.3: Performance comparison between CyRIS and Alfons

Virtual Machine	Disk Image Size [MB]	
	Alfons	CyRIS
Gateway	1387	2048
Dns/mail	3303	3383
File server	4989	5109
Database	4699	4812
Desktop	9987	10227
Total Size	24365	25579
CREATION TIME	**6507.6s**	**793.6s**

totally five virtual machines in the environment: a firewall, a desktop, a file server, a database, and a DNS/mail server.

Note that, however, the Alfons paper, while gives details about the network topology, provides no information about the content inside the training environment. Thus, we assume that the content only includes basic setup for the system, including necessary tools for each server to function properly, and a number of forwarding rules as the means of communication among them. In detail, we put (i) `MySQL Server` for the database, (ii) `Samba` for the file server, (iii) `BIND` and `Postfix` for the DNS/mail server, and (iv) `Iptables` for the firewall server. Besides, we create an administration account in each server, and an user account on the client desktop. The traffic forwarding rules configured in the firewall server including the rules for (i) the client and DMZ segments to reach the database via port 3306, (ii) the client and DMZ segments to reach the file server via port 139 and 445, and (iii) the client segments to reach DMZ segments via port 25 and 53.

4.2.2.2 Experimental Results

Table 4.3 shows the result about our comparison between Alfons and CyRIS. As we can see, even though the total size of our environment is slightly larger, CyRIS is able to construct the complete cyber range in roughly 13 minutes, whereas Alfons takes more than 100 minutes to do so. This result, on one hand, indicates that CyRIS works more efficiently than Alfons in preparing cyber ranges for cyber security training.

However, on the other hand, as having mentioned before, we have no way to clarify if the contents we put in our training environment are matched with what Alfons authors did with theirs. Moreover, they mentioned nothing about how they measure the creation time for the environment. Since 100 minutes is too long, we suspect that this measured time includes the task of creating the base images from scratch. This is different from CyRIS as it constructs the cyber range based on the base images which have been created in advance. Another difference is that for setting up topology in a cyber range, Alfons configures VLAN on switches whereas CyRIS creates virtual interfaces and bridges on physical hosts. Alfons method, although appears efficient in many cases, takes longer due to the delay in response from the switches. In addition, from the detail they give in the paper, we notice that their total amount of time is exactly equal to the sum of the times Alfons spends on creating the cyber range in each physical machine. This, from our perspective, indicates that Alfons does not have an efficient parallel mechanism for creating cyber ranges on multiple hosts. This disadvantage might result to poor performance, or even failure, when it comes to large-scale construction scenarios.

Chapter 5

Conclusion

We started this thesis by surveying available training programs in order to study the current status of cybersecurity training and education. Along the way, we found that using cyber ranges was one of the best ways not only for practicing hands-on skills, but also for improving the training, to make it more interactive and engaging. Nonetheless, we saw the time and effort required to construct such environments manually was high, as very few open-source tools could help, even partially, facilitate this process. We also became well aware that this kind of training is mainly conducted in military environments, and that mostly only proprietary products are available. Given that cyber attacks occur daily basis, we realized that an orchestrating tool, which could be accessed and used by anyone, and would facilitate the job of preparing cyber ranges for cybersecurity training, would be of the utmost importance. Thus, we designed and implemented CyRIS, as an open-source tool, to provide instructors with an automated way in the task of creating cybersecurity training environments.

In some ways, CyRIS is similar to other well-known tools in the automated environment configuration area, as it supports basic functions, such as managing accounts, installing tools, executing scripts, cloning virtual machines, etc., for preparing the general infrastructure needed for a cyber range. The main difference regarding CyRIS, however, lies in a set of security functions for configuring sophisticated security-related content in the cyber range, such as emulating various types of attacks, producing capture traffic data, and emulating dummy malware. These novel security functions offer the instructors a new way of constructing realistic training environments in an efficient and flexible manner. In addition, CyRIS has the ability to setup the network service among virtual machine instances, and is able to isolate the environment from the outside network for safety purposes.

CyRIS is well-suited for use in practical cybersecurity training programs. Our evaluation has shown that by combining different functions, CyRIS is able to prepare the content needed for all the security testing and assessment techniques discussed in the relevant U.S. NIST guideline. Moreover, its execution performance has been proved sufficient enough to meet a reasonable target time for cyber range construction. Our results have shown that it took under 10 minutes to construct an environment for 200 participants, when each of them is provided a VM containing basic setups, and under 15 minutes for a scenario of a class including 100 participants, two machines with more sophisticated security-related content are assigned for each of them.

Beside all the advantages that CyRIS possesses over other similar well-known tools, there exists several incomplete points in its implementation, which need to be improved. The first one is its limitation in terms of host OS support. At the current stage, CyRIS is only able to run on a host using Ubuntu Server as hypervisor (either 14.04 LTS or 16.04 LTS). There are various other OSes available, such as Mac OS, Windows, RedHat, etc., and it is highly important that CyRIS can operate on all of them.

The second limitation of CyRIS is that it can only install security-related content on individual nodes using Centos7 OS. The reason we chose Centos7 for CyRIS to start with, as having said in Chapter 3, is that this is the latest community version of RedHat Enterprise Linux, which is usually the first choice of large companies and organizations for their servers because of the OS performance, stability and security. However, other OSes, such as Windows, often appear in company networks as well, but CyRIS is unable at the moment to install content on the OSes.

Another improvement we would like to have in CyRIS is about IoT security support. With the quick development of IoT technology, people are also now at risk because of IoT security issues. It is worth mentioning that the world "record" for the biggest DDoS attack is 1 Tbps, which was performed mainly from compromised IoT devices [9]. Being aware of this trend, current training programs have started including topics of IoT security in their content. The current implementation of CyRIS only allows installing content and vulnerability in a desktop-like machine, but not in IoT devices, such as Raspberry Pi, Android phone, etc.

With the quick increase in the number of cyber attacks happening nowadays, we believe that cybersecurity training and education become more and more important, as they help strengthen people's knowledge and educate IT security professionals to protect the society. This thesis aims to design and implement CyRIS, an open-source tool which can help facilitate the task of preparing environments in cybersecurity training. Our plan is to release public the source code of CyRIS at the end of this fiscal year (March 2017). We hope that CyRIS will serve as a good means in the future, both for instructors in constructing cyber ranges for their courses, and for individuals in creating custom environments for practicing hands-on activities.

References

[1] The Wall Street Journal - Health Insurer Anthem Hit by Hackers. Retrieved on Dec 22, 2016 from http://www.wsj.com/articles/health-insurer-anthem-hit-by-hackers-1423103720.

[2] Reuters - Japan pension system hacked, 1.25 million cases of personal data leaked. Retrieved on Dec 22, 2016 from http://www.reuters.com/article/us-japan-pensions-attacks-idUSKBN0OH1OP20150601.

[3] SECURITYWEEK NETWORK: Information Security News - Hosting Provider OVH Hit by 1 Tbps DDoS Attack. Retrieved on Dec 22, 2016 from http://www.securityweek.com/.

[4] BBC - South China Sea: Vietnam airport screens hacked. Retrieved on Dec 22, 2016 from http://www.bbc.com/news/world-asia-36927674.

[5] Cyber Security: Protect and Defend. Retrieved on Jan 10, 2017 from https://www.udemy.com/cyber-security/.

[6] Cybrary - Free and Open Source Cyber Security Learning. Retrieved on Jan 10, 2017 from https://www.cybrary.it/.

[7] Web Application Security Forum. Hardening Project (in Japanese). Retrieved on Jan 10, 2017 from http://wasforum.jp/hardening-project/.

[8] Ministry of Internal Affairs and Communications, Japan. Cyber Defense Exercise with Recurrence (CYDER) Training Program (press release). Retrieved on Jan 10, 2017 from http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Releases/Telecommunications/130925_02.html.

[9] World's largest 1 Tbps DDoS Attack launched from 152,000 hacked Smart Devices. Retrieved on Jan 12, 2017 from http://thehackernews.com/2016/09/ddos-attack-iot.html.

[10] Top 7 Network Attack Types in 2016. Retrieved on Jan 25, 2017 from http://www.calyptix.com/top-threats/top-7-network-attack-types-2016/.

[11] R. Beuran, K. Chinen, Y. Tan, and Y. Shinoda. Towards Effective Cybersecurity Education and Training. Technical Report IS-RR-2016-003, Japan Advanced Institute of Science and Technology (JAIST), October 2016.

[12] R. Beuran, C. Pham, D. Tang, K. Chinen, Y. Tan, Y. Shinoda. CyTrONE: An Integrated Cybersecurity Training Framework. International Conference on Information Systems Security and Privacy (ICISSP 2017), Porto, Portugal, February 19-21, 2017.

[13] C. Pham, D. Tang, K. Chinen, R. Beuran. CyRIS: A Cyber Range Instantiation System for Facilitating Security Training. International Symposium on Information and Communication Technology (SoICT ACM 2016), Ho Chi Minh, Vietnam, December 8-9, 2016.

[14] Ansible is Simple IT Automation. Retrieved on March 22, 2016 from http://www.ansible.com.

[15] OpenStack - OpenStack Open Source Cloud Computing Software. Retrieved on March 22, 2016 from https://www.openstack.org.

[16] Development Environments Made Easy. Retrieved on August 3rd, 2016 from https://www.vagrantup.com.

[17] Chef | IT Automation for Speed and Awesomeness | Chef. Retrieved on August 4th, 2016 from https://www.chef.io/chef.

[18] Server Virtualization with VMware vSphere. Retrieved on August 4th, 2016 from http://www.vmware.com/pruducts/vsphere.html.

[19] Toshiyuki Miyachi, Takeshi Nakagawa, Ken-ichi Chinen, Shinsuke Miwa and Yoichi Shinoda. StarBED and SpringOS Architectures and their Performance. 7th International ICST Conference, TRIDENTCOM 2011.

[20] National Institute of Information and Communication Technology, Japan. Hokuriku StarBED Technology Center. Retrieved August 1st, 2016 from http://starbed.nict.go.jp/.

[21] Shingo Yasuda, Ryosuke Miura, Satoshi Ohta, Yuuki Takano, Toshiyuki Miyachi. Alfons: A Mimetic Network Environment Construction System. 11th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, TridentCom 2016.

[22] GitHub - facebook/fbctf: Platform to host Capture the Flag competitions. Retrieved on August 4th, 2016 from https://github.com/facebook/fbctf.

[23] Moodle - Open-source learning platform. Retrieved on August 3rd, 2016 from https://moodle.org/.

[24] Murugiah P. Souppaya, Karen A. Scarfone. Technical Guide to Information Security Testing and Assessment. Special Publication 800-115. National Institute of Standards and Technology, 2008.

[25] Fabrice Bellard. (2011) QEMU - Open Source Processor Emulater homepage. [Online]. Retrieved on November 18th, 2016 from http://www.qemu.org/.

[26] Red Hat, Inc. (2011) Kernel-based Virtual Machine (KVM) homepage. [Online]. Retrieved on November 18th, 2016 http://www.linux-kvm.org/.

[27] SANS Information Security Training | Cyber Certifications | Research. Retrieved on August 4th, 2016 from https://www.sans.org/.

[28] The CERT Division | SEI | CMU. Retrieved on August 4th, 2016 from www.cert.org/.

[29] Netfilter: Firewalling, NAT, and Packet Mangling for Linux - The netfilter.org "iptables" project. Retrieved on December 25th, 2016 from http://www.netfilter.org/projects/iptables/.

[30] Paramiko - A Python implementation of SSHv2. Retrieved on December 25th, 2016 from http://www.paramiko.org/.

[31] HPING 3. Retrieved on December 26th, 2016 from http://www.hping.org/hping3.html.

[32] Ubuntu Manpage: parallel-scp - parallel versions of scp. Retrieved on December 27th, 2016 from http://manpages.ubuntu.com/manpages/trusty/man1/parallel-scp.1.html.

Publications

[1] R. Beuran, **C. Pham**, D. Tang, K. Chinen, Y. Tan, Y. Shinoda, "CyTrONE: An Integrated Cybersecurity Training Framework", International Conference on Information Systems Security and Privacy (ICISSP 2017), Porto, Portugal, February 19-21, 2017.

[2] **C. Pham**, D. Tang, K. Chinen, R. Beuran, "CyRIS: A Cyber Range Instantiation System for Facilitating Security Training", International Symposium on Information and Communication Technology (SoICT 2016), Ho Chi Minh, Vietnam, December 8-9, 2016

[3] D. Tang, **C. Pham**, K. Chinen, R. Beuran, "Interactive Cyber Attack Emulation for Facilitating Security Training", poster, Internet Conference (IC 2016), Tokyo, Japan, October 11-12, 2016.

Appendix A

Cyber Range Description Example

This appendix shows an example of a cyber range description, in which it includes all key words that could be used in CyRIS.

```
#######################################################################
# This is a sample cyber range definition description for the CyRIS
# cyber range instantiation system. The description is written in
# YAML format and contains three main sections, as follows:
# - host_settings:  Contains information about the hosts the cyber
#                   range is to be deployed on, including an id, a
#                   management address, a virtual bridge address,
#                   and a management account.
# - guest_settings: Provides information about the base images. It
#                   defines the entire content of the cyber range
#                   that CyRIS needs to prepare.
# - clone_settings: Gives details about the cloning phase. It has a
#                   unique range id, and information about hosts,
#                    guests, and network topology.
#######################################################################
#######################################################################
# Start of the host_settings section. For each host, it consists of 4 fields:
#  @param    id           The id of the host.
#  @param    mgmt_addr    The management address used to access the host.
#  @param    virbr_addr   The virtual bridge address that the host uses
#                         to communicate with the virtual machines that
#                         will be created on it. On installation KVM
#                         typically creates a default virtual bridge
#                         virbr0 with default IP address 192.168.122.1.
#  @param    account      The account name on the host that is to be
#                         used by CyRIS. It accesses the host by SSH
#                         public key, which needs to be prepared in
#                         advance by the user.
#
# The number of hosts specified in this section is not limited. In this
# example, it consists of one host with id "host_1".
#######################################################################
- host_settings:
  - id: host_1
    mgmt_addr: 172.16.24.1
    virbr_addr: 192.168.122.1
    account: cyuser
#######################################################################
# Start of the guest_settings section. For each base image, it consists
# of 7 fields:
# @param    id                  The id of the base image, which by
#                               convention reflects its role in the
#                               cyber range (desktop, web server,
#                               database, etc.).
# @param    ip_addr             The IP address of the base image, so
#                               so that CyRIS can access it from
#                               the host.
# @param    basevm_host         The host that the base image is
#                               located on.
# @param    basevm_config_file  The path to the configuration file
#                               of the base image, for KVM to be able
#                               to define and start it.
# @param    basevm_type         The virtualization type of the base
#                               image. Currently CyRIS supports only
#                               KVM virtualization.
# @param    tasks               The definition of all the setup tasks
#                               that CyRIS needs to do on the base
#                               image. Specific parameters needed for
#                               each kind of task are presented in the
#                               section below.
#
# The number of base images specified in this section is also not limited.
# This example includes three of them, a desktop, a web server, and a
# firewall server.
#######################################################################
- guest_settings:
  - id: desktop
    ip_addr: 192.168.122.50
    basevm_host: host_1
    basevm_config_file: /home/cyuser/basevm_desktop_dev.xml
    basevm_type: kvm
    basevm_name: basevm_desktop_dev
    tasks:
    # The add_account task is for adding a new account to the base image.
    # It needs two parameters, which are:
    # @param    account      Name of the new account.
    # @param    passwd       Password of the new account.
    - add_account:
      - account: daniel
        passwd: JamesBond
    # The modify_account task is for changing the information of an
    # existing account. Depending on the need to modifying only the
    # name, only the password, or both, it needs two or three parameters,
    # which are:
    # @param    account        Name of an existing account.
    # @param    new_account    New name for the specified account.
    # @param    new_passwd     New password for the specified account.
    - modify_account:
      - account: root
        new_passwd: theroot
    # The install_package account task is for installing a package
    # from a repository, by using some package manager (yum, apt-get,
    # and so on, depending on the OS of the base image). It has three
    # parameters:
    # @param    package_manager     yum, apt-get, etc., depending
    #                               on the OS.
    # @param    name                Name of the package to be installed.
    # @param    version             Version of the package (optional). By
    #                               default CyRIS will install the latest
    #                               version.
    - install_package:
      - package_manager: yum
        name: wireshark
        version: 1.8.10
      - package_manager: yum
        name: GeoIP
    # The emulate_attack task is for deploying an attack in the cyber range
    # , which is used to create logs and other traces. In the current stage,
    # CyRIS supports emulating actual ssh dictionary attacks. The needed
    # parameters are presented below:
    # - SSH dictionary attack:
    #   @param    attack_type      Type of the attack (ssh_attack).
    #   @param    target_account   The account that is targeted in the attack.
    #   @param    attempt_number   Number of login attempts in the attack.
    - emulate_attack:
      - attack_type: ssh_attack
        target_account: daniel
        attempt_number: 200
    # The emulate_traffic_capture_files task is for generating pcap files
    # that contain specific network traffic traces. In the current stage,
    # CyRIS supports generating traffic traces for ssh dictionary attack,
    # DOS attack, and DDOS attack.
    # For each traffic pattern, it needs different parameters, as follows:
    # - SSH dictionary attack:
    #   @param    format            Format of the traffic file (pcap).
    #   @param    filename          Name of the traffic file.
    #   @param    attack_type       Type of the traffic pattern (ssh_attack).
    #   @param    attack_source     The IP address from which the attack is
    #                               initiated.
    #   @param    noise_level       How much normal traffic needs to be added
    #                               into the file, to make it more challenging
    #                               to detect the attack pattern. The level can
    #                               be low, medium, high.
    # - DOS attack and DDOS attack:
    #   @param    format            Format of the traffic file (pcap).
    #   @param    file_name         Name of the traffic file.
    #   @param    attack_type       Type of the traffic pattern (dos_attack,
    #                               ddos_attack).
    #   @param    noise_level       How much normal traffic needs to be mixed
    #                               into the file, so as to make it more
    #                               challenging to detect the attack pattern.
    #                               The level can be low, medium, high.
    - emulate_traffic_capture_files:
      - format: pcap
        file_name: /home/trainee/traffic1.pcap
        attack_type: ssh_attack
        attack_source: 2.95.120.235
        noise_level: medium
      - format: pcap
        file_name: /home/trainee/traffic2.pcap
        attack_type: dos_attack
        noise_level: medium
    # The emulate_malware task is for starting a dummy malware process. It can
    # run under two modes at the current stage: performing a calculation or
    # listening to a specified port. Depending on the execution mode, the task
    # takes different parameters:
    # - Dummy calculation mode:
    #   @param    name               Name of the dummy malware process.
    #   @param    mode               Running mode of the malware (dummy_calculation).
    #   @param    cpu_utilization    Percent of CPU the malware will consume.
    #
    # - Port listening mode:
    #   @param    name               Name of the dummy malware process.
    #   @param    mode               Running mode of the malware (port_listening).
    #   @param    port               Port number the malware listens to.
    - emulate_malware:
      - name: spyeye
        mode: dummy_calculation
```

```
      cpu_utilization: 40
    # The copy_content task is for copying some data (files, etc.) to the cyber
    # range. This task needs two parameters:
    # @param    src    The absolute path (on the host) of the data to be copied.
    # @param    dst    The absolute path (on the guest) of the location where to
    #                  copy the data to.
    - copy_content:
      - src: /home/cyuser/cyris-development/database/penetration_testing
        dst: /bin/cyberrange
      - src: /home/cyuser/cyris-development/database/flag.txt
        dst: /root
    # The execute_program task is for executing scripts/programs (python, bash,
    # ruby, etc.), which have been copied to the cyber range in the copy_content
    # tasks. This function needs three parameters:
    # @param    program        The absolute path of the location of the program.
    # @param    interpreter    The required shell/interpreter/compiler (python,
    #                          bash, ruby, gcc, etc.).
    # @param    execute_time   This parameter is optional, indicating the time
    #                          that CyRIS will executes the program, whether
    #                          before or after cloning phase. If users want to
    #                          execute it before the cloning phase, then this
    #                          parameter is not needed to specified. In case of
    #                          after-cloning-phase execution, the value for it
    #                          should be "after_clone".
    - execute_program:
      - program: /bin/cyberrange/database/penetration_testing/install_pip.sh
        interpreter: bash
      - program: /bin/cyberrange/database/penetration_testing/prepare.sh
        interpreter: bash
        execute_time: after_clone
    # The firewall_rules task is for specifying and setting up firewall rules.
    # CyRIS will automatically set up the firewall rules specified in the
    # provided file(s). Only the iptables tool is currently supported.
    # @param    rule    The file which contains a list of firewall rules.
    - firewall_rules:
      - rule: /home/cyuser/database/firewall_ruleset_forwarding
      - rule: /home/cyuser/database/firewall_ruleset_inputoutput
#######################################################################
# Specify information for the base image named "webserver".
- id: webserver
  ip_addr: 192.168.122.51
  basevm_host: host_1
  basevm_config_file: /home/cyuser/kvm/basevm_webserver.xml
  basevm_type: kvm
  basevm_name: basevm_webserver_dev
  tasks:
    - add_account:
      - account: daniel
        passwd: JamesBond
    - install_package:
      - package_manager: yum
        name: httpd
    - emulate_traffic_capture_files:
      - format: pcap
        file_name: /home/traffic.pcap
        attack_type: ddos_attack
        noise_level: medium
  ########################################################################
  # Specify information for the base image named "firewall".
  - id: firewall
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/gateway.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: robot.abc
        passwd: abcrb1357
########################################################################
# Start of the clone_settings section. All the settings are done inside
# one field:
# @param   range_id    The unique ID of the cyber range, which is used
# to differentiate between cyber ranges created on the same host.
########################################################################
- clone_settings:
  - range_id: 112
    # hosts is for specifying a list of hosts that the cyber range is
    #                            deployed on.
    # @param    host_id          The ID of the host (as specified in the
    #                            host_settings section located in the
    #                            beginning of this file).
    # @param    instance_number  The number of instances of the cyber range
    #                            that are going to be deployed on the host.
    # @param    guests           Specify a list of virtual machines which
    #                            are cloned from the base images included
    #                            in the guest_settings section.
    # @param    topology         The network topology among virtual machines
    #                            in the cyber range.
    hosts:
    - host_id: host_1
      instance_number: 2
      # guests is for specifying a list of virtual machines which are cloned
      # from the base images specified in the guest_settings section. For
      # each guest, it takes several parameters:
      # @param    guest_id        The ID of the base image that the virtual
      #                           machine is cloned from.
      # @param    number          The number of the virtual machines that
      #                           will be cloned.
      # @param    firewall_rules  The forwarding rules for communication
      #                           done between network segments. This field
      #                           is optional, and it is only specified
      #                           when the virtual machine plays the role of
      #                           forwarding traffic in the network (such as
      #                           a firewall machine, gateway machine, etc.).
      guests:
      # Specify the virtual machines which are to be cloned from the
      # "desktop" base image. No firewall rules are included for these
      # virtual machines.
      - guest_id: desktop
        number: 1
      # Specify the virtual machines which are to be cloned from the
      # "webserver" base image. No firewall rules are included for these
      # virtual machines.
      - guest_id: webserver
        number: 1
      # Specify the virtual machines that are to be cloned from the
      # "firewall" base image.
      - guest_id: firewall
        number: 1
        forwarding_rules:
        # These rules are for forwarding traffic in the network. They
        # contain several parameters:
        # @param    src      The source segment for the traffic. If
        #                    there are multiple source segments, they
        #                    are separated by the character ",".
        # @param    dst      The destination segment of the traffic.
        #                    Same "," separation rule is applied when
        #                    multiple destinations are specified.
        # @param    sport    The source port of the traffic.
        # @param    dport    The destination port of the traffic.
        - rule: src=office,external_servers dst=internal_servers.database dport=3306
        - rule: src=office,external_servers dst=internal_servers.fileserver dport=139
        - rule: src=office dst=external_servers dport=25,53
      # Topology is for the network topology among virtual machines in
      # the cyber range. It has several parameters:
      # @param    type      The type of the network in the cyber range
      #                     (custom type, ring type, dumbbell type, etc.).
      #                     At the current stage, only the custom type is
      #                     supported.
      # @param    networks  A list of network segments in the cyber range,
      #                     including name of the segment, members, and
      #                     gateway.
      topology:
      - type: custom
        # The list of network segments in the cyber range.
        # @param    name       Name of the network segment.
        # @param    members    A list of members in the network segment.
        #                      They are specified as
        #                      <guest_id>.<network_interface> values.
        # @param    gateway    A gateway for members in this segment to
        #                      communicate with other segments.
        networks:
        # A network segment named office.
        - name: office
          members: desktop.eth0
          gateway: firewall.eth0
        # A network segment named servers.
        - name: servers
          members: webserver.eth0
          gateway: firewall.eth1
```