CyRIS User Guide

March 2019

Cyber Range Organization and Design
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, 923-1292 Japan
https://www.jaist.ac.jp

CyRIS User Guide

Contents

1 Introduction

This is the user guide for CyRIS, an open-source software tool for automatically creating and managing environments used in cybersecurity training, called *cyber ranges*; the name of the software is an acronym for "Cyber Range Instantiation System". CyRIS is being developed by the Cyber Range Organization and Design (CROND) NEC-endowed chair at Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan. This user guide refers to CyRIS v1.1, which was released in March 2019.

1.1 Overview

CyRIS is mainly intended for assisting cybersecurity education and training instructors in preparing the necessary hands-on environments; for a research background, please consult the reference [5]. CyRIS is the core component of the integrated cybersecurity training framework CyTrONE that is also being developed by CROND; for more information, please refer to [4].

1.1.1 Working flow

The working flow of CyRIS is described in Figure 1.1. The two inputs that the system takes to create the desired cyber range are:

- A cyber range description file;
- A set of virtual machine base images.

The cyber range description file is for instructors to describe the composition and content of the cyber range. It can be created manually or generated from a template. This description is written in YAML, a text-based file format, and defines all the necessary information for creating a given cyber range. Chapter 2 discusses in detail how to create such a file.

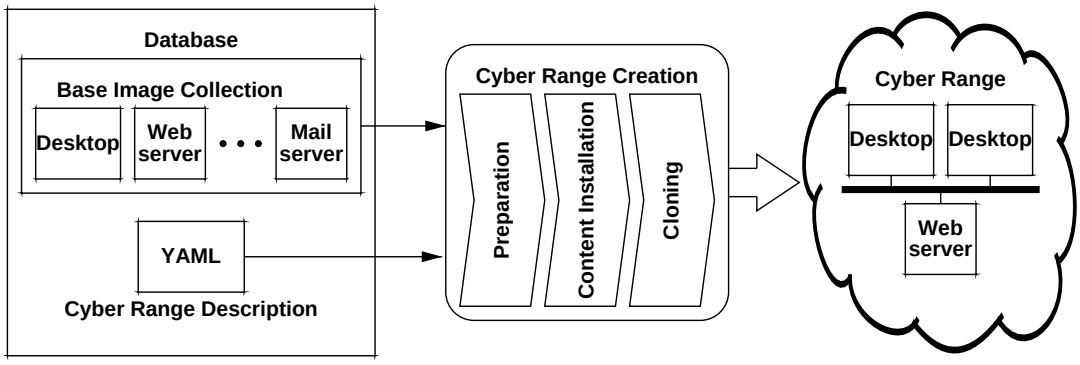

Figure 1.1: The working flow of CyRIS.

The base images are used via the KVM virtualization system, and they contain a pre-installed operating system and several basic system configurations (e.g., IP address, etc.). More information about how to prepare a base image for CyRIS is provided in Chapter 3.

1.1.2 System architecture

The overall system on which CyRIS is running has the architecture presented in Figure 1.2. The execution infrastructure is represented by a collection of hosts, each of them equipped with a virtualization platform (currently QEMU/KVM), which are connected to a LAN network.

One of the CyRIS hosts is designated as a *master host*, and manages the entire execution, performing tasks such as: processing the input description file, preparing the base images and installing security content into them, and cloning the virtual machines to other hosts.

1.1.3 Functionality

In order to create security-related content in the cyber range, CyRIS offers a wide range of functions, as summarized in Table 1.1. These functions are divided into two categories, as basic functions and security functions. The first group includes functions that are commonly found in other well-known automated environment configuration tools (Ansible, Chef, etc.), such as installing packages, copying data, configuring network service, and so forth. The security functions represent the main difference in CyRIS compared to other systems, and they include functions for reproducing actual security-related incidents, such as attacks, traffic capturing, and so on, in order to generate corresponding content.

1.1.4 Changes

The current CyRIS release focuses mainly on portability improvement. The most important modification is support for Windows in guest VM [7]. See the file `CHANGES` that is

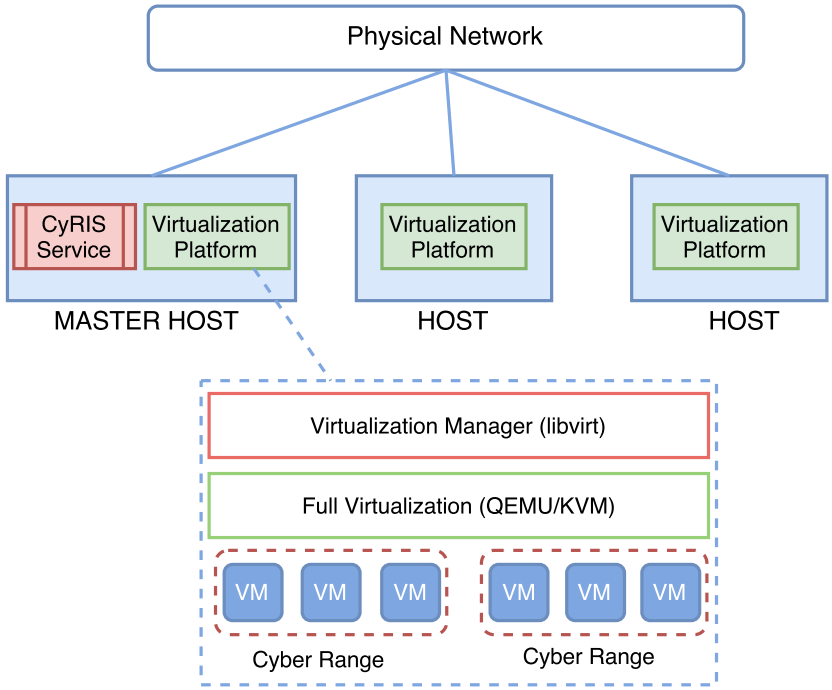

Figure 1.2: Overall architecture of the system on which CyRIS is running.

Table 1.1: Overview of CyRIS functionality

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	

included with the release for a more complete list.

1.2 Document structure

This user guide is structured as follows. In Chapter 2 we explain the YAML description used for defining the content in the target cyber range. In Chapter 3 we present the practical use of CyRIS, including instructions on how to conduct experiments. The document ends with references and an appendix.

1.3 Another Document

Document [8] describes more widely information than this guide. It mentions other CROND products (CyLMS, CyTrONE and its web-GUI). However it was written by Japanese and care only v1.0 .

1.4 Contact information

CyRIS is being released via GitHub, which should be used exclusively for download and issue reporting. The link to the repository is given below:

`https://github.com/crond-jaist/cyris`

For any other inquiries or suggestions, either about the present user guide, or about CyRIS itself, you can contact us via e-mail at the following address:

`crond-sec(AT)jaist.ac.jp`

2 Cyber range description

This chapter describes the syntax of the cyber range description file used as a CyRIS input, which is written using the YAML format [1]. For parsing the description file we use the PyYAML library for the Python programming language [2]. The description is divided into three sections, as explained below:

- `host_settings` contains information about the hosts the cyber range is to be deployed on, including an id, a management address, and an account;
- `guest_settings` provides information about the base images used to instantiate the cyber range, and defines all the content of the cyber range that CyRIS needs to prepare;
- `clone_settings` gives details about the cloning phase, and contains an unique range id, along with other information, such as the actual hosts that the cyber range is to be deployed on, the number of guests, and the network topology between them.

Although in the next sections we shall provide several examples of how to use each of the cyber range description parameters, we also recommend checking the files in the `examples/` directory of the distribution, in particular `full.yml`, for practical use cases.

2.1 `host_settings` section

The first part of the cyber range description file is `host_settings`, which includes information about the hosts that will be involved in the cyber range deployment. The first host specified in this section is designated as the master host, which processes the description file, and creates the cyber range across itself and the other hosts. The parameters users need to specify for each host, are presented next, together with their roles, expected data types and values.

`id` The id of the host, used internally to refer to it.

- Data type: string
- Value: any value

mgmt_addr The management address of the host, used to communicate with it.

- Data type: string
- Value: any value

virbr_addr The address of the KVM virtual bridge on the host[1].

- Data type: string (IPv4 address representation)
- Value: any value (default = `192.168.122.1`)

account The account on the host that will be used by CyRIS.

- Data type: string
- Value: any value

The number of hosts specified in this section is not limited. Below is an example of a description consisting of two hosts, in which the first of them, with id `host_1`, becomes the master host.

```
- host_settings:
  - id: host_1
    mgmt_addr: 172.16.24.1
    virbr_addr: 192.168.122.1
    account: cyuser
  - id: host_2
    mgmt_addr: 172.16.24.2
    virbr_addr: 192.168.122.1
    account: cyuser
```

2.2 guest_settings section

The second and main part of the description file is `guest_settings`, which provides information about the base images and content installation. Each guest section starts with global information about the guest and the base image used to create it (such as guest id, base image file, etc.). This is followed by a set of operations that need to be performed on it in order to create the desired guest content. This section explains first about the parameters for specifying global information regarding a guest, then goes into detail about how to use CyRIS functions to perform various operations for setting it up.

[1]Usually it appears on the host as an interface with the name `virbr0`, and its default IP address is `192.168.122.1`.

2.2.1 Global information

The necessary global information parameters for a guest, their roles, expected data types and values are as follows.

id The id of the guest, which by convention reflects its role in the cyber range (desktop, web server, database, and so on).

- Data type: string
- Value: any value

ip_addr An optional parameter for specifying the base image IP address used during the configuration process of this guest. If not provided, CyRIS automatically assigns a default IP address for the base image [2].

- Data type: string (IPv4 address representation)
- Value: 192.168.122.N (with N between 2 and 254)

basevm_host Specifies the id of the host on which the base image is located. In the current implementation, CyRIS only supports base images that are located on the master host.

- Data type: string
- Value: master host id

basevm_config_file KVM uses an XML configuration file to define a base image, which provides necessary information, such as RAM, CPUs, NICs, etc. This parameter specifies the location of the XML configuration file.

- Data type: string (an absolute file path)
- Value: any value

basevm_type Specifies the type of the base image; CyRIS currently supports only KVM virtualization, but other technologies, such as Docker containers or VMWare, may be supported in the future.

- Data type: string
- Value: `kvm`

basevm_os_type Specifies the OS type of the base image; See also Section 2.5.

- Data type: string
- Value: { `windows.7` }

[2] Currently the convention is as follows: for the nth base image in the `guest_settings` section (starting from 0), the default IP address is generated as `192.168.122.`X, where $X = 100 + n$.

tasks Specifies a list of operations that will be performed on the base image. Details about these tasks will be given in the next sections.

The number of guests in this section is also not limited. Below is an example of a description consisting of two guests called `desktop` and `webserver`. Note that the `desktop` guest is specified first, and the parameter `ip_addr` is not included, hence its IP address will be assigned the value `192.168.122.100`. Similarly, the `webserver` will be assigned the IP address `192.168.122.101`.

```
- guest_settings:
  - id: desktop
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_desktop.xml
    basevm_type: kvm
    tasks:
      ...
  - id: webserver
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_webserver.xml
    basevm_type: kvm
    tasks:
      ...
```

CyRIS also provides a number of functions, which help instructors to perform various operations on a base image, so as to create their desired security-related content. These functions are divided into five main categories, which are *system configuration*, *tool installation*, *incident emulation*, *content management*, *clone management* (cf. Table 1.1).

The first four categories are for creating independent content on each guest, and are specified within the subsection `tasks` of the section `guest_settings` in the description file. The last function group is about cloning and setting up network service, thus it is specified in the section `clone_settings` (see Section 2.3).

The remainder of this section describes the setup operations that the instructors can ask CyRIS to perform on the guest base images within the `tasks` subsection.

2.2.2 `add_account` task

This function is used for creating a new account on the base image. The necessary parameters, their roles, expected data types and values are as follows.

account Specifies the user name of the account to be created.

- Data type: string
- Value: any value

passwd Specifies the password of the account to be created.

- Data type: string
- Value: any value

full_name Optional parameter for specifying the full name of the user.

- Data type: string
- Value: any value

The number of new accounts to be created is not limited. Below is an example of a description for creating a new account on the guest "desktop" (the global information is omitted). The new account has the name "daniel", the password is "danielpass", and the user's full name is "Daniel Radcliffe".

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - add_account:
      - account: daniel
        passwd: danielpass
        full_name: Daniel Radcliffe
```

2.2.3 modify_account task

This function is for editing information of existing accounts on the base image. The necessary parameters, their roles, expected data types and values are as follows.

account Specifies the user name of the account being modified.

- Data type: string
- Value: name of an existing account

new_account Specifies a new name for the account when the intention is to change the account name.

- Data type: string
- Value: any value

new_passwd Specifies a new password for the account when the intention is to change the account password.

- Data type: string
- Value: any value

Depending on the need to modify only the account name, only the password, or both, this operation needs only two or all the three parameters above. Below is an example of a description for modifying the password of the "root" account to "abcd1234" on the guest "desktop" (the global information is omitted).

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - modify_account:
      - account: root
        new_passwd: abcd1234
```

2.2.4 `install_package` task

This function is for installing packages on the base image. The necessary parameters, their roles, expected data types and values are as follows.

`package_manager` Optional parameter for specifying a package management tool to be used to install the desired package. If this parameters is not provided, `yum` will be used as package manager.

- Data type: string
- Value: { `yum` | `apt-get` | `chocolatey` }

`name` Specifies the name of the package to be installed.

- Data type: string
- Value: any value

`version` Optional parameter for specifying the package version. By default, CyRIS will install the latest version of the package available in the repository.

- Data type: string
- Value: any value

Below is an example description for installing the package called "wireshark" with version "1.8.10" on the guest "desktop". Note that since the default value for the package manager is "yum", the corresponding line could have been omitted.

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - install_package:
      - package_manager: yum
        name: wireshark
        version: 1.8.10
```

2.2.5 emulate_attack task

This function is for emulating an attack on the base image. Currently CyRIS supports three kinds of attacks, namely SSH dictionary, DoS and DDoS attacks, and this functionality is available on the CentOS 7 operating system only. The first type of attack is done via stand-alone emulation, since logs and traces are consequently generated automatically in the system. For the latter two attacks, one needs to add other tasks for capturing the packet traces, as it will be discussed in Section 2.2.6.

This section describes the necessary parameters, their roles, expected data types and values for an SSH dictionary attack emulation.

attack_type Specifies the attack type. As mentioned above, SSH dictionary attack is the only option at this moment.

- Data type: string
- Value: `ssh_attack`

target_account Specifies the account on the base image that the attack should target.

- Data type: string
- Value: name of an existing account

attempt_number Specifies how many attempts of the attack should be performed. Note that the bigger the attempt number, the longer the emulation takes.

- Data type: int
- Value: any value

attack_time Optional parameter for specifying the date when the attack is supposed to have taken place. If this information is provided, CyRIS modifies the system date accordingly before starting the attack emulation. Otherwise, the current date will be used.

- Data type: string
- Value: $YYYY\text{-}MM\text{-}DD$ (or $YYYYMMDD$)

Below is an example of a description for an SSH dictionary attack which targets the account named "daniel" on the base image "desktop". The attack includes 54 attempts, and is considered to have taken place on "November 18th, 2017".

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - emulate_attack:
      - attack_type: ssh_attack
        target_account: daniel
        attempt_number: 54
        attack_time: 2017-11-18
```

2.2.6 `emulate_traffic_capture_file` task

This function is for emulating network sniffing activities, and capturing traffic in PCAP format, e.g., to include attack traces. In the current stage, CyRIS supports generating traffic traces for SSH dictionary, DoS and DDoS attacks. Below are the necessary parameters, their roles, expected data types and values.

`format` Specifies the format of the traffic capture file. Currently only the PCAP format is supported.

- Data type: string
- Value: `pcap`

`file_name` Specifies the absolute path of the file the traffic capture should be written to.

- Data type: string
- Value: any value

`attack_type` Specifies the type of the attack from among the supported ones.

- Data type: string
- Value: { `ssh_attack` | `dos_attack` | `ddos_attack` }

`attack_source` Specifies the IP address from where the attack is supposed to have been initiated. This parameter is only necessary for the `ssh_attack`.

- Data type: string (IPv4 address representation)
- Value: any value

`noise_level` Specifies how much normal traffic "noise" needs to be added into the capture file, for the purpose of making it more challenging to detect the attack pattern.

- Data type: string
- Value: { `low` | `medium` | `high` }

Below is an example of a description for emulating an SSH dictionary attack and a DDoS attack. The traces are written to the files "traffic1.pcap" and "traffic2.pcap", respectively, and the noise levels are "medium" in both cases. The SSH attack is supposed to originate from the IP address "69.89.31.226".

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - emulate_traffic_capture_file:
      - format: pcap
        file_name: /home/trainee/traffic1.pcap
        attack_type: ssh_attack
        attack_source: 69.89.31.226
```

```
    noise_level: medium
  - format: pcap
    file_name: /home/trainee/traffic2.pcap
    attack_type: ddos_attack
    noise_level: medium
```

2.2.7 emulate_malware task

The malware emulation function is used to run a dummy malware process, which has no actual malicious behavior. The corresponding command is placed in the file `/etc/rc.d/rc.local`, so that it is run automatically as root when the OS boots.

The malware emulation functionality currently implemented in CyRIS has two distinct execution modes:

1. Perform a calculation to increase the CPU load, similar to the symptoms of some of the malicious processes;
2. Listen to a specified port, so as to reproduce the behavior of a malware that opened a backdoor.

Below are the necessary parameters for configuring the malware emulation mode, their roles, expected data types and values.

name Specifies the name the dummy malware should use.

- Data type: string
- Value: any value

mode Specifies the mode for the dummy malware from among the supported ones.

- Data type: string
- Value: { `dummy_calculation` | `port_listening` }

cpu_utilization Specifies the percentage of CPU time that the malware will consume when using the `dummy_calculation` mode.

- Data type: integer
- Value: [0, 100]

port Specifies the number of the port the malware will listen to in case the malware is configured to use the `port_listening` mode.

- Data type: integer
- Value: [1, 65535]

Below is an example of a description for emulating two malware processes. The first one, called “daemon” uses the “dummy_calculation” mode, and will cause a 40% CPU utilization. The second one, called “spyeye”, uses the “port_listening” mode, and will listen on port “1052”.

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - emulate_malware:
      - name: daemon
        mode: dummy_calculation
        cpu_utilization: 40
      - name: spyeye
        mode: port_listening
        port: 1052
```

2.2.8 `copy_content` task

The copy content function is for copying data, both files and directories, from an external location to the base images, for instance in order to create a desired configuration, place files for trainees to investigate, etc. Below are the necessary parameters, their roles, expected data types and values.

`src` Specifies the absolute path *on the master host* of the file/directory to be copied.

- Data type: string
- Value: any

`dst` Specifies the absolute path *on the base image* of the location where the data is to be copied to.

- Data type: string
- Value: any

Below is an example of a for copying the directory "penetration_testing" and the text file "flag.txt" from the master host to the base image. The destinations are the directories "/bin/cyberrange" and "/home/trainee", respectively.

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - src: /home/cyuser/database/penetration_testing
      dst: /bin/cyberrange
    - src: /home/cyuser/database/flag.txt
      dst: /home/trainee
```

2.2.9 `execute_program` task

The execute program function is for executing scripts/programs, such as Python, bash, or Ruby scripts, on the base image. The programs must already exist before executing this task, for instance by a copy operation as discussed above. Below are the necessary parameters, their roles, expected data types and values.

program Specifies the absolute path of the program location.

- Data type: string
- Value: any

args Optional parameter for specifying any arguments that are needed to run the above program. Arguments should be separated by spaces.

- Data type: string
- Value: any

interpreter Specifies an interpreter for running the program from among the supported ones.

- Data type: string
- Value: { `python` | `bash` | `ruby` }

execute_time Optional parameter indicating whether the program should be executed *after* the cloning phase, otherwise it is not necessary. This is useful in case the execution depends on the properties of an actual virtual machine instance in the final cyber range, such as its IP address, etc.

- Data type: string
- Value: `after_clone`

Below is an example of a description for executing two bash scripts. The first one, "install_tool.sh", is executed *before* the cloning phase to install a certain tool in the guest base image; it also requires two arguments, "/tool/path" and "parameter". The second script, named "prepare.sh", will be executed *after* the cloning phase, for instance to do some machine-dependent configuration.

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - execute_program:
      - program: /bin/cyberrange/penetration_testing/install_tool.sh
        args: /tool/path parameter
        interpreter: bash
      - program: /bin/cyberrange/penetration_testing/prepare.sh
        interpreter: bash
        execute_time: after_clone
```

2.2.10 `firewall_rules` task

The firewall rule operation is for specifying firewall rules; CyRIS will automatically set up the firewall rules specified in the provided file(s). Only the `iptables` firewall program is currently supported. Below are the necessary parameters, their roles, expected data types and values.

rule Specifies the absolute path of the file containing the list of firewall rules to be configured.

- Data type: string
- Value: any

Below is an example in which two files are used to define firewall rules, one for forwarding actions, and the other one for I/O operations. The files are named "firewall_ruleset_forwarding" and "firewall_ruleset_input_output", respectively.

```
- guest_settings:
  - id: desktop
    ... <global information> ...
    tasks:
    - firewall_rules:
      - rule: /home/cyuser/database/firewall_ruleset_forwarding
      - rule: /home/cyuser/database/firewall_ruleset_input_output
```

2.3 `clone_settings` section

The third main section in the cyber range description file is `clone_settings`, which gives details about the cloning phase. Each cyber range has a unique id, along with other information, such as the hosts that cyber range is deployed on, the guests which would be included in the cyber range, and the network topology. The necessary parameters that users need to specify in this section are presented next, together with their roles, expected data types and values.

range_id Specifies a unique id of the cyber range, which is used to differentiate between cyber ranges created on the same host. Currently, this id is used as the first byte in the network address of the cyber range, therefore one must avoid values that cannot be assigned to the first byte in an IP address.

- Data type: integer
- Value: [1, 254]

hosts Specifies the hosts, described as a YAML list, that the cyber range is to be deployed on. The following parameters should be provided for each host.

host_id The id of the host the cyber range should be deployed on. This host id must be already defined in the section `host_settings` of the description file.

- Data type: string
- Value: an existing host id

instance_number The number of cyber range instances of this type that are to be deployed on the host. Note that cyber range performance can decrease if too many instances are deployed on a given host.

- Data type: integer
- Value: any positive value

guests A list of guests that are to be cloned from the descriptions provided in the `guest_settings`. The following parameters should be indicated for each guest.

guest_id The id of the guest that the virtual machine is to be cloned from.

- Data type: string
- Value: an existing guest id

number The number of virtual machines of this type that are to be cloned.

- Data type: integer
- Value: any positive value

forwarding_rules An optional list of forwarding rules for communication among network segments. This parameter is only specified when a guest plays the role of forwarding traffic in the network (such as a firewall machine, gateway machine, etc.). Each rule must contains four parameters: `src` and `dst` for the source and destination segment, and `sport` and `dport` for the source and destination port of the traffic.

- Data type: string
- Value: `src=`SRC `dst=`DST `sport=`$SPORT$ `dport=`$DPORT$

entry_point An optional keyword indicating whether the current guest is to be used as entry point for accessing the cyber range[3]. One and only one occurrence of this keyword is necessary.

- Data type: string
- Value: `yes`

topology Specifies the network topology among the virtual machines in the cyber range given as a list of networks. The following parameters should be specified for each network.

type The type of the network in the cyber range. At the current stage, only a "custom" type is supported, but "ring" type or "dumbbell" type specifications can also be imagined.

- Date type: string
- Value: `custom_type`

networks A list of network segments in this cyber range network, including the name of the segment, a list of guest members, and a gateway, as follows.

name The name of the network segment, such as "office", "internal", "external", etc.

- Data type: string

[3]For each cyber range instance, CyRIS will set up an SSH tunnel, so that trainees can connect from outside the cyber range to the designated entry point guest.

- Value: any value

members A comma-separated enumeration of members in this network segment. For each member one should specify the guest id and its network interface, connected by a period character, for example, "desktop.eth0", where "desktop" is a guest id, and "eth0" is an interface name.

- Data type: string
- Value: `guest_id.interface`, `guest_id.interface`, ...

gateway Optional parameter to indicate the default gateway for members in this segment. If this parameter is given, then the IP address of the indicated guest/interface will be set as the default gateway for all the members in the segment. Otherwise, the default gateway address is set automatically by appending `1` to the three first bytes of the segment network address, e.g., `<byte1>.<byte2>.<byte3>.1`.

- Data type: string
- Value: `guest_id.interface` (default is explained above)

The following should be taken into account regarding the `clone_settings` section of the cyber range description file:

- As mentioned above, the `range_id`, is used not only to differentiate among cyber ranges, but also as the first byte in the network address of the cyber range. Therefore, it is important to assign it an appropriate value;
- Due to the complex network topology a cyber range might have, and also to avoid human mistakes, the IP addresses for individual virtual machines in the cyber range are assigned automatically by CyRIS as follows:
  - The network address for the entire cyber range: `<range_id>.1.1.1/8`
  - The network address for a cyber range instance: `<range_id>.<instance_id>.1.1/16` (with `instance_id` starting from 1)
  - The network address of each network segment, depending on the order in which the segments are defined in the description file: `<range_id>.<instance_id>.<segment_no>.1/24` (with the value of `segment_no` starting from 1)
  - The IP address of each cloned virtual machine in a segment, depending on the order in which it appears in the `members` enumeration of the `networks` subsection of `topology` in `clone_settings`: `<range_id>.<instance_id>.<segment_no>.<member_no>` (the value of `member_no` starts from 2, and it also depends on how many guests of a certain type are specified in the `number` field of the `guests` subsection of `hosts` in `clone_settings`)

– The default gateway IP address of every member in a network segment (if the `gateway` parameter in the `networks` subsection of `topology` in `clone_settings` is not specified):
`<range_id>.<instance_id>.<segment_no>.1`

Below is an example of a description for a cyber range with id "112" that is to be deployed on host "host_1". A number of 2 instances of this cyber range will be deployed, each of them consisting of 2 "desktop" machines and 1 "firewall" one, which is also designated as the entry point to the instance. The desktops are all connected via their interface "eth0" to the same network segment, named "office", and they will use the "eth0" interface IP address of the "firewall" machine as the default gateway IP address.

```
- host_settings:
  - id: host_1
    ...<host info>...

- guest_settings:
  - id: desktop
    ...<guest info>...
  - id: firewall
    ...<guest info>...

- clone_settings:
  - range_id: 112
    hosts:
    - host_id: host_1
      instance_number: 2
      guests:
      - guest_id: desktop
        number: 2
      - guest_id: firewall
        number: 1
        entry_point: yes
      topology:
      - type: custom
        networks:
        - name: office
          members: desktop.eth0
          gateway: firewall.eth0
```

Considering the IP address assignment rules that we have introduced above, the IP address of each machine in the cyber range will be configured as follows:

- **Cyber range 112, instance #1**:
  - **desktop #1**:
    - IP address: `112.1.1.2`
    - Default gateway: `112.1.1.4`
  - **desktop #2**:
    - IP address: `112.1.1.3`
    - Default gateway: `112.1.1.4`

  - **firewall**:
    - IP address: `112.1.1.4`
    - Default gateway: `112.1.1.1`

- **Cyber range 112, instance #2**:
  - **desktop #1**:
    - IP address: `112.2.1.2`
    - Default gateway: `112.2.1.4`
  - **desktop #2**:
    - IP address: `112.2.1.3`
    - Default gateway: `112.2.1.4`
  - **firewall**:
    - IP address: `112.2.1.4`
    - Default gateway: `112.2.1.1`

2.4 Detailed example

Here we provide an example of a complete description for creating a realistic cyber range. The target setup has a DMZ network topology as shown in Figure 2.1, and the actual CyRIS description for this cyber range in included below the figure.

The main characteristics of the DMZ cyber range, as indicated in the main sections of the description, are as follows:

- `host_settings`: The cyber range is to be deployed on a single physical host, which is given the id "host_1", followed by management and virtual bridge IP addresses and account information;

- `guest_settings`: A total of five guests are defined: "firewall", "dnsmail", "filesrv", "dbsrv", and "desktop". For each guest the base image properties are provided, followed by a series of setup tasks, such as adding accounts, installing packages, copying files, and executing scripts;

- `clone_settings`: A single cyber range instance will be created on the host, which will contain one guest of each type. For the "firewall" guest some forwarding rules are defined, and it is designated as the entry point for connecting to the cyber range instance. The network topology is composed of 3 segments: one for external servers, one for internal servers, and one for office, each with the corresponding guests attached to them.

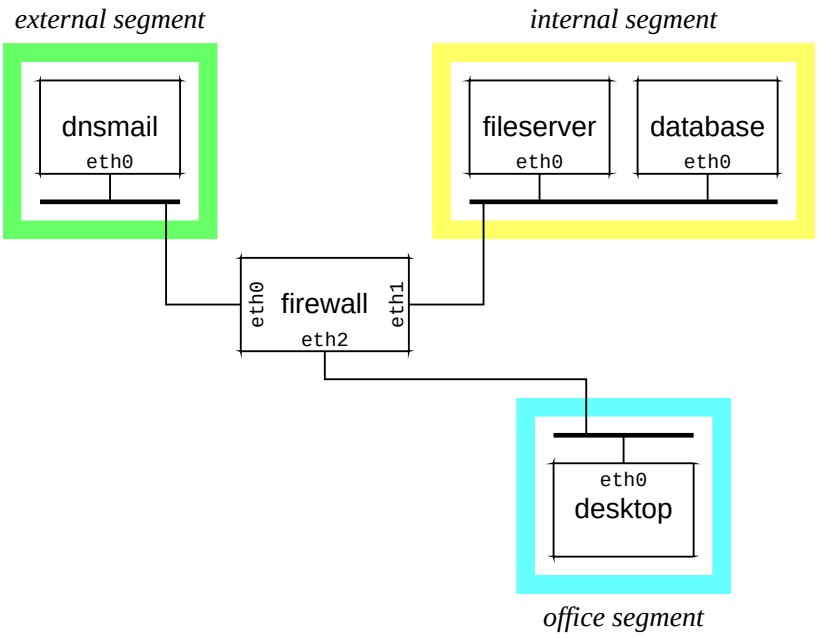

Figure 2.1: Target DMZ network topology for the example description.

```
---
- host_settings:
  - id: host_1
    mgmt_addr: 172.16.1.7
    virbr_addr: 192.168.122.1
    account: cyuser

- guest_settings:
  - id: firewall
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_firewall.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: robot.abc
        passwd: abcrb1357
    - modify_account:
      - account: root
        new_passwd: abcd.1234

  - id: dnsmail
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_dnsmail.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: robot.abc
        passwd: abcrb1357
```

```
    - modify_account:
      - account: root
        new_passwd: abcd.1234
    - install_package:
      - package_manager: yum
        name: wget
      - package_manager: yum
        name: telnet

  - id: filesrv
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_filesrv.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: robot.abc
        passwd: abcrb1357
    - modify_account:
      - account: root
        new_passwd: abcd.1234
    - install_package:
      - package_manager: yum
        name: samba samba-client samba-common
      - package_manager: yum
        name: wget

  - id: dbsrv
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_dbsrv.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: robot.abc
        passwd: abcrb1357
    - modify_account:
      - account: root
        new_passwd: abcd.1234
    - install_package:
      - package_manager: yum
        name: wget

  - id: desktop
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm_desktop.xml
    basevm_type: kvm
    tasks:
    - add_account:
      - account: daniel
        passwd: JamesBond
    - install_package:
      - package_manager: yum
        name: nmap
      - package_manager: yum
        name: telnet

- clone_settings:
  - range_id: 124
    hosts:
    - host_id: host_1
      instance_number: 1
      guests:
      - guest_id: firewall
```

```
    number: 1
    forwarding_rules:
    - rule: src=office,external dst=internal.dbsrv dport=3306
    - rule: src=office,external dst=internal.filesrv dport=139,445
    - rule: src=office dst=external dport=25,53
    entry_point: yes
  - guest_id: dnsmail
    number: 1
  - guest_id: filesrv
    number: 1
  - guest_id: dbsrv
    number: 1
  - guest_id: desktop
    number: 1
 topology:
 - type: custom
   networks:
   - name: external
     members: dnsmail.eth0
     gateway: firewall.eth0
   - name: internal
     members: filesrv.eth0, dbsrv.eth0
     gateway: firewall.eth1
   - name: office
     members: desktop.eth0
     gateway: firewall.eth2
```

2.5 Remarks

CyRIS is still under active development, and its functionality is continuously extended. Table 2.1 shows a summary of its current features, along with the extend of their support for several guest VM operating systems on which we have focused so far, CentOS(7), Ubuntu(16.04 LTS and 18.04 LTS) and Windows(7). Although only the CentOS 7 operating system is fully supported at this moment, we are currently working on extending CyRIS support to other OSes.

CyRIS v1.1 supports Windows 7. Since Windows are **commercial products**, you have to care its license. **We assume no responsibility for license.** If you want to use Windows 7, set `windows.7` into `basevm_os_type` in cyber range description (see page.7 .) Otherwise, do not describe `basevm_os_type` — CyRIS v1.1 will work like v1.0 . We may change or expand around this parameter in the future to support various OSes.

The CyRIS distribution includes several sample files in the directory called `examples/`, and we recommend checking them in advance in order to better understand the description file syntax. Here is a brief overview of each sample file:

- `basic.yml`: Simple cyber range without additional guest settings, to be deployed on a single host;
- `basic_multi-host.yml`: Same simple cyber range, this time which is to be deployed on two hosts;
- `dmz.yml`: A complex cyber range with five hosts and a DMZ topology (this is the cyber range already discussed in Section 2.4);

Table 2.1: Current CyRIS support depending on guest OS type

Functionality	CentOS 7	Ubuntu 16.04 LTS, 18.04 LTS	Windows 7
Add accounts	✓	✓	✓
Modify accounts	✓	✓	✓
Install packages	✓	✓	✓
Emulate attacks	✓		
Emulate traffic capture files	✓		
Emulate malware	✓		
Copy content	✓	✓	✓
Execute programs	✓	✓	✓
Modify firewall rules	✓		
Clone VMs	✓	✓	✓

- `full.yml`: A commented example that showcases all the functionality of CyRIS, mainly intended as a convenient reference for its features.

Next we present several clarifications regarding the information provided in the previous sections. Some of the restrictions mentioned here apply to the current version of CyRIS, but may change in future versions:

- The base image indicated in the base image configuration file specified in the global information area of a guest via the keyword `basevm_config_file` in `guest_settings` should be in the same directory with the configuration file itself. Moreover, both files must have the name, with no extension for the base image, and the ".xml" extension for the configuration file.

- The noise addition feature for traffic capture file generation specified via the `emulate_traffic_capture_file` task requires that several PCAP files are available in the directory `instantiation/logs_preparation`. Due to anonymization issues, we do not include such files in the CyRIS distribution. If you require this functionality, please create them by capturing traffic with various data rates, and name the files as follows:
`noise_low.pcap`, `noise_medium.pcap`, `noise_high.pcap`

- The tool `cpulimit`[4] is currently used to control CPU utilization for dummy malware emulation specified via the `emulate_malware` task, but due to license issues it is not included in the CyRIS distribution. If you require this functionality, please copy its *source code* into the following directory:
`instantiation/malware_creation/cpulimit`

[4]The tool `cpulimit` is available here: `https://github.com/opsengine/cpulimit`

- Default gateway configuration by CyRIS, either using the default rule or via the keyword `gateway` in the `networks` subsection of the `topology` section in `clone_settings` only sets the default gateway IP address on the corresponding guests, but does not provide any actual forwarding functionality. When using the default configuration mechanism, the default address is actually that of the virtual bridge connecting the corresponding network segment to the physical host, therefore it represents no actual gateway machine. Consequently, if forwarding functionality is required, users should employ the `gateway` keyword, and enable packet forwarding on the target guest.

3 Utilization

This chapter provides information on how to employ CyRIS to create and managing environments for cyber security training courses.

3.1 Hardware requirements

To run CyRIS, your physical machine should have the Ubuntu OS installed. We have tested CyRIS running on Ubuntu Server 14.04 LTS and 16.04 LTS. In addition, the KVM virtualization platform should be installed, and your physical hosts must have a processor that supports hardware virtualization. To verify this, please follow instructions from the KVM Installation page [3], "Check that your CPU supports hardware virtualization" section.

It is also recommended to have an Internet connection available on the master host, so that CyRIS can connect to Linux repositories for downloading and installing packages during the cyber range creation process. If your master host has no Internet access, please make sure to set up local repositories instead.

3.2 Installation

The following steps should be carried out in order to install CyRIS, as it will be explained next:

1. Host preparation;
2. Base image preparation;
3. CyRIS configuration.

3.2.1 Host preparation

After downloading the CyRIS source code, please install it on the master host (e.g., in `/home/cyuser/cyris`). Several tasks then need to be done on the physical hosts intended for CyRIS, as described below:

- The account used to run CyRIS must be granted sudo execution permission without entering a password. This is because CyRIS drives all the screen output during the creation process to a log file, and users will not know when they are asked to input the sudo password.

- SSH keys must be generated, e.g., by running the command `ssh-keygen` (please refer to [6] for more information). In this context, note the following:

  - When using multiple hosts for cyber range instantiation, such as in the example discussed in Section **??**, *all the hosts* involved must use the same pair of private & public SSH keys. This is required so that the virtual machines deployed on those hosts can be accessed in a unified manner without password. Moreover, the user account on the master host should also be able to login to the other hosts without password, so that it can distribute the base images that are to be cloned.
  - When using a gateway between the master host and the Internet, the user needs to setup key-based SSH authentication from the master host to the gateway (e.g., by using the command `ssh-copy-id`). Otherwise, CyRIS will ask for the gateway password when it tries to create tunnels for cyber range access.

- Several external tools need to be installed on the hosts. For automating this task, a bash script called `HOST_PREPARE.sh` is given (see the `cyris/` directory). Please make sure to execute this script successfully on all the hosts before running CyRIS.

In the scenario when a cyber range is deployed on multiple hosts, we recommend to copy in advance the CyRIS source code to all of them and to the *same location*, as well as to use the *same account name* for CyRIS operation on all hosts.

Some KVM-specific settings are also required, please perform them if necessary:

- The user account employed when running CyRIS must be part of the `libvirtd` group. Otherwise, an error such as the one given below may occur:
  ```
  libvirtError:  Failed to connect socket to
  '/var/run/libvirt/libvirt-sock':  Permission denied
  ```

- The `libvirt-qemu` account used internally by KVM must be able to access the directories in which the cyber range is being created. Otherwise, a "permission denied" error will occur during the instantiation process.

3.2.2 Base image preparation

At the current stage, CyRIS fully supports base images for CentOS 7 guests, and partially Ubuntu 16.04 LTS ones. A base image archive is provided via the CyRIS GitHub repository for testing purposes, and contains two files: the disk image file of a sample VM, and the associated XML configuration file for KVM. Please download the archive and extract its content somewhere on the master host (e.g., into the directory `/home/cyuser/images`).

Users can also create their own base images; please refer to Appendix for more details. As mentioned already in Section 2.5, when using CyRIS some constraints are currently imposed on the file names and location: (i) the file name of the base VM disk image should have no extension at all (e.g., "basevm"); (ii) the associated KVM configuration file should be named in the same way with the base image, but with the extension ".xml" (e.g."basevm.xml"); (iii) both the base image and its configuration file should be located in the same directory[1].

Note that CyRIS needs in principle to connect to the Internet for certain tasks of the cyber range instantiation process, such as package installation. Therefore, the following should be considered regarding the base image:

- If the CyRIS host has a direct Internet connection, the DNS settings of the base VM should be checked (e.g., the file `/etc/resolv.conf`). The sample VM uses the default KVM bridge address as DNS setting, but you may need to change this depending on your setup;

- If the CyRIS host connects to the Internet via a proxy server, then appropriate proxy settings have to be configured in the base VM at least for package installation (e.g., via the file `/etc/yum.conf` for `yum`); depending on the cyber range content, global proxy settings may also be required;

- If the CyRIS host cannot connect to the Internet, for instance due to security reasons, then you will need to set up alternative package installation methods, such as using a local repository, etc.

3.2.3 CyRIS configuration

CyRIS uses a configuration file to set some of its internal parameters. An example is provide in the file `cyris/CONFIG` (see Figure 3.1). The parameters are:

`cyris_path` The absolute path of the directory where CyRIS is located.

`cyber_range_dir` The absolute path of the directory where CyRIS stores data related to the cyber ranges that are being created; this data will be removed upon cyber range destruction.

[1]Note that, given that the XML configuration file contains explicitly the location of the base image file, it needs to be updated if the base VM files are moved.

gw_mode Indicate whether there is a gateway host that stands between the physical hosts and the outside network. If so, then gw_mode should be set to on or true. In contrast, if the physical hosts connects directly to the outside network, then gw_mode should be set to off or false.

gw_account The user account on the gateway host.

gw_mgmt_addr The management address of the gateway host.

gw_inside_addr The internal address of the gateway host, which is used for communicating with the physical hosts.

Please note that if gw_mode = off, then gw_account, gw_mgmt_addr, and gw_inside_addr are ignored and do not need to be set.

```
[config]

# The absolute path of the top CyRIS directory
# (remember to have the slash "/" at the end of the path)
cyris_path = /home/cyuser/cyris/

# The absolute path where cyber ranges are to be instantiated
# (remember to have the slash "/" at the end of the path)
cyber_range_dir = /home/cyuser/cyris/cyber_range/

# Information regarding the gateway
# (details are not used if gw_mode is set to "off")
gw_mode = off
#gw_account = gw_user
#gw_mgmt_addr = gw_hostname
#gw_inside_addr = 172.16.1.1
```

Figure 3.1: Example configuration file for CyRIS.

3.3 Cyber range instantiation

Once the installation steps have been performed, and the content of the cyber range has been specified in a description file, users can run CyRIS as follows:

```
> <cyris_path>/main/cyris.py <description_file> <cyris_path>/CONFIG
```

where <cyris_path> is the absolute path to CyRIS, and <description_file> is the name of the cyber range description file.

The files related to a given cyber range that are generated during the creation phase, such as log files, temporary base images, etc., are all stored in the subdirectory range_id of the cyber range directory specified in the CONFIG file, where range_id is the cyber range id specified in the clone_settings section of the input cyber range description

file. For example, if a cyber range has the range id "123", then its directory will be: `/home/cyuser/cyris/cyber_range/123/`.

Two key files generated during the instantiation process are:

- `creation.log`: A detailed log of the cyber range creation activities;
- `cr_creation_status`: Notification of the final execution status, determined based on a built-in verification mechanism: `SUCCESS` or `FAILURE`[2].

Several command-line options are available for CyRIS, as shown below:

```
USAGE: cyris.py [options] RANGE_DESCRIPTION CONFIG_FILE

OPTIONS:
-h, --help              Display help
-d, --destroy-on-error  In case of error, try to destroy cyber range
-v, --verbose           Display verbose messages for debugging purposes
```

If CyRIS execution finishes successfully, two files will be created in the corresponding cyber range directory `<cyber_range_dir>/<range_id>/`, as follows:

- `range_notification-cr<range_id>.txt`: Information about how to login to the cyber range (via an automatically generated SSH tunnel to the entry point guest);
- `range_details-cr<range_id>.yml`: Network details about the created cyber range.

Please consult these files before accessing the cyber range. Note that only the login information needs to be provided to trainees, the details are only intended for the instructor.

3.4 Cyber range destruction

The recommended way to perform cyber range destruction is to use the program called `range_cleanup.py`, and provide the id of the cyber range to be destroyed as argument, together with the CONFIG file:

```
> <cyris_path>/main/range_cleanup.py <range_id> <cyris_path>/CONFIG
```

where `<cyris_path>` is the absolute path to CyRIS, and `<range_id>` is the id of the cyber range to be destroyed.

Note that, during the cyber range creation phase, a script is generated by CyRIS for the purpose of destruction in the corresponding cyber range directory, and is named `whole-controlled-destruction.sh`. This script is used internally by the `range_cleanup.py`

[2]Given that the error-checking mechanism is insufficient in some cases, some errors may still appear even if the execution status is `SUCCESS`, so manual checking may be needed occasionally.

program, and could also be executed manually if desired. The program, however, is able to clean up the cyber range even if the destruction script is not created, e.g., because of an error during the instantiation process.

In some cases, especially after the cyber range creation process crashes repeatedly, one may want to destroy *all* cyber ranges and clean up all temporary files. In this situation, please execute the script `<cyris_path>/destroy_all_cr.sh`, with two arguments: the absolute path of CyRIS, and the cyber range directory (as they were specified in the `CONFIG` file).

Bibliography

[1] Get Started - YAML Ain't Markup Language. Retrieved on Jan 17, 2017 from http://www.yaml.org/start.html.

[2] PyYAML.org - the home of various YAML implementations for Python. Retrieved on Jan 17, 2017 from http://pyyaml.org/.

[3] KVM/Installation. Retrieved on Jan 31, 2017 from https://help.ubuntu.com/community/KVM/Installation.

[4] 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, pp. 157-166.

[5] 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, pp. 251-258.

[6] How To Set Up SSH Keys. Retrieved on March 17, 2017 from https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2.

[7] Yuichiro Sakamoto, Kobayakawa Michihiro, Chinen Ken-ichi and Beuran Razvan: *Impovement of CyRIS for Building Various Kinds of Cyber Training Instances*, Cyber Resilience Conference (CRC2018), pp.29, Putrajaya, Malaysia, 2018.

[8] Gen Komatsu: *CROND Cyber Security Training System v1.0 Install Guide*, IS-TM-2018-003, JAIST Repository, ISSN 0918-7561, 2018 (Japanese).

`http://hdl.handle.net/10119/15514`

A Guest VM Base Image Preparation

A.1 CentOS 7 base image preparation

Assuming that you have created a CentOS 7 base image for KVM, there are some steps that need to be taken in order for it to operate correctly with CyRIS:

1. **Network interface configuration**: In CentOS 7, there is no `ifconfig` command. Moreover, the network interface names have been changed from `eth*` to `ens*`. The current implementation of CyRIS uses the `ifconfig` command, and traditional interface names. Follow these steps to get them back in CentOS 7:

   (a) Install `net-tools` via `yum` so that `ifconfig` is available

   (b) Change `ens*` to `eth*` names[1]:

      i. In the file `/etc/default/grub`, look for the line that starts with `GRUB_CMDLINE_LINUX`, and append the text below:
      `net.ifnames=0 biosdevname=0`

      ii. Execute the command below:
      `grub2-mkconfig -o /boot/grub2/grub.cfg`

   (c) Remove any file such as the one below, if it exists:
   `/etc/sysconfig/network-scripts/ifcfg-ens?`

2. **IP address configuration**: CyRIS uses the MAC address included in KVM configuration files to store the IP address the corresponding virtual machine should have after booting. For this mechanism to work properly, the next actions should be taken:

   (a) Copy the following directory to the base image:
   `cyris/instantiation/vm_clone/initif`

[1] Reference: http://unix.stackexchange.com/questions/81834/how-can-i-change-the-default-ens33-network-device-to-old-eth0-on-fedora-19

(b) Configure the start on boot of the `initif` script by adding the following command into the file `/etc/rc.d/rc.local`:
`<path>/initif/initif <path>/initif/initif.conf`

(c) Make the `rc.local` file executable (if it is not already):
`chmod +x /etc/rc.d/rc.local`

3. **Firewall configuration**:

(a) Install `iptables-services` via `yum`, and delete all the existing rules:
```
yum -y install iptables-services
systemctl enable iptables
iptables -F
iptables -t nat -F
service iptables save
```

(b) Add the following line in the file `/etc/rc.d/rc.local` file to enable the firewall setup:
`iptables-restore /etc/sysconfig/iptables`

4. **Additional configuration**:

(a) Disable the NetworkManager by executing the commands below:
```
systemctl stop NetworkManager
systemctl disable NetworkManager
```

(b) In case your physical hosts use a proxy to connect to Internet, insert the proxy information into the appropriate files in the base image. For instance, adding the proxy into `/etc/yum.conf` is mandatory for being able to install packages via `yum`. Depending on the cyber range content, global proxy settings in `/etc/environment`, or specific proxy settings (for example, in `/etc/wgetrc` for `wget`) may be required.

(c) Depending on your setup, DNS settings may be required, e.g., via the file `/etc/resolv.conf`. We recommend a typical setting for KVM, such as `192.168.122.1`, which is the default IP address used for the KVM bridge[2].

(d) Set the root password to "theroot". This is needed so that CyRIS can connect to the VM the very first time. The password can be changed if desired when the cyber range is created via the `modify_account` task in the `guest_settings` section.

(e) Reboot the guest and check that all settings are as expected (recommended).

A.2 Ubuntu 16.04 LTS base image preparation

For Ubuntu 16.04 LTS base images too several procedures are needed for them to operate correctly with CyRIS:

[2]To reduce the effects of potentially invalid DNS settings on ssh connections, you can also set "UseDNS no" in `/etc/ssh/sshd_config`.

1. **Network interface configuration**: In Ubuntu 16.04 LTS as well, the network interface names have been changed from `eth*` to `ens*`. As the current implementation of CyRIS uses traditional interface names, follow these steps to get them back in Ubuntu 16.04 LTS:

   (a) Change `ens*` to `eth*` names:

      i. In the file `/etc/default/grub`, look for the line that starts with `GRUB_CMDLINE_LINUX`, and append the text below:
      `net.ifnames=0 biosdevname=0`

      ii. Execute the command below:
      `sudo update-grub`

   (b) Remove any configuration lines for old interfaces in the file below:
   `/etc/network/interfaces`

2. **IP address configuration**: CyRIS uses the MAC address included in KVM configuration files to store the IP address the corresponding virtual machine should have after booting. For this mechanism to work properly, the next actions should be taken:

   (a) Copy the following directory to the base image:
   `cyris/instantiation/vm_clone/initif`

   (b) Configure the start on boot of the `initif` script by adding the following command into the file `/etc/rc.local`:
   `<path>/initif/initif <path>/initif/initif.conf`

   (c) Make the `rc.local` file executable (if it is not already):
   `chmod +x /etc/rc.local`

3. **Additional configuration**:

   (a) Disable the NetworkManager by executing the commands below:
   `systemctl stop NetworkManager.service`
   `systemctl disable NetworkManager.service`

   (b) Enable SSH login as root by editing the file `/etc/ssh/sshd_config`:
   `PermitRootLogin yes`

   (c) In case your physical hosts use a proxy to connect to Internet, insert the proxy information into the appropriate files in the base image. For instance, adding the proxy into `/etc/apt/apt.conf` is mandatory for being able to install packages via `apt`. Depending on the cyber range content, global proxy settings in `/etc/environment`, or specific proxy settings (for example, in `/etc/wgetrc` for `wget`) may be required.

   (d) Depending on your setup, DNS settings may be required, e.g., via the file `/etc/resolv.conf`. We recommend a typical setting for KVM, such as

192.168.122.1, which is the default IP address used for the KVM bridge[3].

(e) Set the root password to "theroot". This is needed so that CyRIS can connect to the VM the very first time. The password can be changed if desired when the cyber range is created via the modify_account task in the guest_settings section.

(f) Reboot the guest and check that all settings are as expected (recommended).

A.3 Windows 7 base image preparation

This section describes how to make base image of guest VM using Windows.

Requirements

You have to satisfy following conditions to run Windows on CyRIS.

1. Upgrade .NET Framework to 4.5 or higher
2. Upgrade PowerShell to 5.0 or higher
3. Install OpenSSH
4. Install and configure initif
5. Install chocolatey
6. Change username root, and password theroot

Windows7 Setup

This subsection describes configuration to satisfy requirements previous mentioned.

Upgrade .NET Framework to 4.5 or higher

Check the version of .NET Framework You can check the version of .NET Framework by following methods:

- Microsoft's page

  (English) https://support.microsoft.com/en-us/help/318785/how-to-determine-which-versions-and-service-pack-levels-of-the-microso

  (Japanese) https://support.microsoft.com/ja-jp/help/318785/how-to-determine-which-versions-and-service-pack-levels-of-the-microso

[3]To reduce the effects of potentially invalid DNS settings on ssh connections, you can also set "UseDNS no" in /etc/ssh/sshd_config.

- @IT

  (Japanese) http://www.atmarkit.co.jp/ait/articles/1210/26/news086.html

Here, we check them by first way. Open "regedit" and HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET␣Framework␣Setup\. You can see file of each versions. You can confirm the version by see the value of key-"version".

.NET Framework is independent per conversion group. We show categories of that groups [1].

- 1.0
- 1.1
- 2.0,3.0,3.5
- 4,4.5.x,4.6.x,4.7.x

Install .NET Framework 4.6.1 Here, we install .NET Framework 4.6.1. It is current version at December 15th, 2017[2].

1. Download installer from following:

   https://www.microsoft.com/ja-jp/download/details.aspx?id=49981

2. Execute downloaded file
3. Update if you find new items in "Windows Update"

Upgrade PowerShell to 5.0 or higher

Check the version of PowerShell Use following command in PowerShell to check its version. PSVersion in its output is the version of that.

Version of PowerShell

```
$PSVersionTable
```

Moreover, there is another way using following. You can find that in Version in its output.

Version of PowerShell

```
$HOST
```

Install PowerShell 5.0 Install PowerShell 5.0[3] like following:

1. Download installer. In this page shows various OS. Choose your OS.
   `https://www.microsoft.com/en-us/download/details.aspx?id=50395`

2. Execute file downloaded

3. Update if you find new items in "Windows Update"

Install OpenSSH

Use OpenSSH for Windows published in GitHub[4]. Download zip file included the software in following site.

Download URL of OpenSSH

32bits
`https://github.com/PowerShell/Win32-OpenSSH/releases/download/0.0.24.0/OpenSSH-Win32.zip`
64bits
`https://github.com/PowerShell/Win32-OpenSSH/releases/download/0.0.24.0/OpenSSH-Win64.zip`

Download PowerShell script from following site into last downloaded directory. `https://github.com/motya1121/setup_win_openssh/blob/master/setup_win_openssh.ps1`

Finally, execute the file. If the execution is fail, it may the policy of execution is wrong. Change its policy by following command:

```
Set-ExecutionPolicy Unrestricted
```

After install, reboot the machine.

As soon as possible, check connectivity to the machine via SSH, when the machine is up. When you can connect, you can remove downloaded file and script.

Install and configure `initif`

CyRIS resolves IP address by conversion of MAC address. `initif` do its conversion. You can configure them by following:

1. Download zip file from page by following URL
   `https://github.com/motya1121/initif/releases/tag/v0.1`

2. Expand files from downloaded file

3. Move following files into `C:\CyberRange\initif`. If the folder is not exist, make that.
   - initif_input_win.ps1
   - initif_set_win.ps1
   - initif_shape_win.exe
   - initif_win.ps1
   - log.txt
   - initif.conf
4. Confirm `log.txt` is empty.
5. Confirm the content of `initif.conf` is same as following:

```
eth0 192.168.122.0/24 ZZ:ZZ
```

6. Since execute by specification of filename, open `gpedit.msc`.
7. Open [Computer Configuration]-[Windows Setting]-[Scripts]-[Startup]
8. Open tab-`PowerShell Scripts` and entry `initif_win.ps1` by push button-`Add`.
   c.f.) ScriptName: C:\CyberRange\initif\initif_win.ps1
9. Note current MAC address.
10. Reboot the machine.
11. Confirm that 4th octet of IP address is last octet of MAC address previous noted. Care decimal and hexdecimal.

Install chocolatey

CyRIS employ software-`chocolatey` to install software into windows. The software install a software by one command. Install steps are following:

1. Open PowerShell by administrator privilege.
2. Execute command in following site:

Install chocolatey

for command prompt)
`https://chocolatey.org/install#install-with-cmdexe`
for PowerShell)
`https://chocolatey.org/install#install-with-powershellexe`

3. No error is printing, the installation is complete.

Chocolatey's version is printed, when you type `choco` at PowerSell and command prompt. Its help is printed when you type `choco -?`, also.

Change username `root` and password `theroot`

Because CyRIS expects user which name `root` and its password `theroot`, you have to fit them. The program expects administrator privilege, also. See following page to change them:
`https://answers.microsoft.com/ja-jp/windows/forum/windows_7-security/windows-7/6c44b1c6-773a-4cc4-a647-0a7749574cc3?auth=1`

Optional configuration

This subsection described are not required. Configure them if your platform is matched.

Install software probably used Various software is installed on WindowsPC at corporation and family. You may want install such software to make Windows base image.
Following programs are installed certainly:

- WWW browser; Google Chrome
- PDF viewer; Adobe Reader
- Office suite; LibreOffice

You can install such software in bulk by following command:

install typical software by chcolatey

```
choco install -y libreoffice adobereader googlechrome
```

Change keyboard mapping as Japaense Change keyboard maapping as Japanese keyboard[5].

1. Open "Control Panel"
2. Open "System and Security"
3. Open "System の Device Manager"
4. Double click [Keyboards] - [Standard PS/2 Keyboard]
5. Choice tab-"Driver" and push button-"Update Driver"
   - Choice "Browse my computer for driver software"
   - Choice "Let me pick from a list of device drivers on my computer"

- Out check of “Show compative hardware”
- Choice “Manufacturer” to “(Standard Keyboards)” and “Model” to “Japanese PS/2 Keyboard (106/109 Key)”. And push “Next”.
- push “Yes“, when “Update Driver Warning” is appeared.
- push “Close”, when update is done.

6. Push “Close”.
7. Push “Yes“, when screen “System Setting Change” is appeard.
8. After reboot, confirm that keyboard map is changed.

Configuration of Japanese input To support Japanese input, do following steps[6].

1. Open “Control Panel”
2. Open “Clock, Language, and Region”
3. Open “Change display language”
   - Choice tab-“Formats”, change “Format:” to “Japanese(Japan)”.
   - Choice tab-“Location”, change “Current Location” to “Japan”.
   - Choice tab-“Keyboards and Languages”, and open “Change keyboards...”
     - Choice tab “General” and open “Add”.
     - Check [Japanese]-[Keyboard]-[Microsoft IME] and push “OK”.
     - Change “Default input language” to “Japanese (Japan) - Microsoft IME”
     - Choice “English” in “Installed services”, and delete by pushing “Remove”.
     - Push “OK”.
   - Choice tab “Administrative” and open “Change system locale...”
   - Choice “Apply” in diagram to confirm.
   - Choice “Japanese(Japan)” in diagram titled “Region and Language Settings”
   - Choice “Restart now” in diagram to restart.
4. After reboot, Open “[Control Panel] - [System and Security] - [Windows Update]”
5. Open “OO optional updates art available” located the bottom of “Install updates for your computer”.
6. Check “Japanese Languages Pack - Windows7 Service Pack1” and push “OK”.
7. Push “Install updates”.

Change display language to Japaense Change display language to Japanese.

1. Open “Control Panel”;
2. Open “Clock, Language, and Region”
3. Open “Change display language”

Show hidden files

1. Open “Controll Panel” and choice “Apperance and Personalization”
2. Choice “Folder Options” and choice tab ‘View‘.
3. In “Advance settings‘, choice “Show hidden files, folders and drives‘ and push “OK”.

Bibliography

[1] .NET Framework のバージョン分け http://www.atmarkit.co.jp/ait/articles/1211/16/news093.html(Japanese)

[2] .NET Framework 4.5 のインストール https://qiita.com/busonx/items/acb54f852962426eb2ed(Japanese)

[3] PowerShell5.0 のインストール https://qiita.com/busonx/items/2ebca42866833516c772(Japanese)

[4] Windows 用の OpenSSH https://github.com/PowerShell/Win32-OpenSSH(Japanese)

[5] 日本語キーボードへの対応 http://www.sakyou.com/ManualShop/Windows/06_keyboard/02_106keyboard/17_Win7PRO/index.html(Japanese)

[6] 英語版 Windows7 を日本語化する http://eng-notebook.com/blog-entry-82/(Japaense)