CyExec*: Automatic Generation of Randomized Cyber Range Scenarios

Ryotaro Nakata[a] and Akira Otsuka[b]
Institute of Information Security, Yokohama, Kanagawa, Japan

Keywords: Information Security Education, Cyber Security, Cyber Range Scenario, Randomization, Docker Container.

Abstract: With the development of information technology, the need for information security education is increasing, and the effectiveness of cyber range exercises is attracting attention. The cyber range is a system to learn knowledge and skills by experiencing an incident scenario reproduced in a virtual environment. Many scenarios are required to train a security expert through various incident experiences. However, scenario development requires highly specialized expertise. Thus, in practice, only a limited number of scenarios are worn out around. Identical scenarios may decrease the educational effect since the other teams' actions or write-ups on the internet will hint the students. We propose CyExec*, a cyber range system that automatically generates multiple scenarios based on DAG(Directed Acyclic Graph)-based scenario randomization. Multiple scenarios with the same learning objectives can enhance teaching effectiveness and prevent cheating. We developed the DAG-based scenario randomization technique on a Docker-based cyber range system called CyExec. By taking full advantage of Docker's system/network configuration power, we can randomize complex scenarios across multiple networks. Comparison with the VM-based scenario generators, CyExec* outperforms, especially in storage usage. Further, CyExec* only consumes 1/3 memories, 1/4 CPU loads, and 1/10 storage usages. Thus, Cyexec* can operate approximately 3-times more complex scenarios than VM-based systems.

1 INTRODUCTION

With the development of information technology and the internet, cyber-attacks and malware damage worldwide, and information security measures have become necessary everywhere. As a result, the training of information security personnel is very active, and in recent years, skills training by cyber range has received much attention (Maki et al., 2020).

The cyber range is a system that allows students to learn knowledge and skills by experiencing real security incident scenarios in a virtual environment(Vincent E Urias, 2018). The cyber range scenarios provide a highly realistic security incident experience. Therefore, it is expected to be highly educational. However, scenario development and environmental preparation require people with expertise. Many scenarios are required to address a large amount of learning content and repetition, but it is difficult for teachers and institutions to develop them independently (Beuran et al., 2019).

Also, the cyber range exercises are divided into teams of several people each. If a same scenario is offered to all teams, and the learning content is the same, there is a risk of leaking hints while the teams work on the problem at the same place. So it is necessary to prepare scenarios with different content for each group, but this is easily said than done. It is possible to develop multiple scenarios with the same learning objectives, but that would be a huge burden in terms of development (Razvan et al., 2017).

Moreover, existing cyber range consists of large numbers of virtual machines to run scenarios. Depending on the number of attendees and teams, this can require 100 or more virtual instances. The problem is that this requires high-performance hardware that can withstand multiple virtual machines' operation (Maki et al., 2020).

As described above, the existing cyber range is required to develop many scenarios, and preparing an environment to do so will be a significant operational burden (Schreuders et al., 2015). We developed a new cyber range platform, CyExec*, which incorporates random components into cyber range scenarios and leverages containerized virtualization to solve these problems. Not only does this provide many scenario patterns, but it is also a versatile cyber range that is light, fast, and can be used in the cloud.

[a] https://orcid.org/0000-0001-8885-848X
[b] https://orcid.org/0000-0001-6862-2576

Nakata, R. and Otsuka, A.
CyExec*: Automatic Generation of Randomized Cyber Range Scenarios.
DOI: 10.5220/0010324502260236
In *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021)*, pages 226-236
ISBN: 978-989-758-491-6

Table 1: Exercise environments used for information security education.

Type	CTF	Lab Work	cyber range
Target	Beginners- highly skilled Hackers	Students and professionals studying information security	Security personnel and various other roles
Require Skill	Web/Network/ Programing/Encryption etc...	Specific Vulnerabilities and attack methods	Comprehensive response to security incidents
Scenario contents	One-off scenarios in game format	A single scenario to address a specific vulnerability	Ability to respond comprehensively to the entire incident response
Implementation cost	Low-Medium (Large-scale events are also held)	Medium (A hands-on, primarily on a personal device)	High (Reproduce real-world equivalent systems in a virtual environment)
Examples	PicoCTF SecGen	Webgoat Labtainer	Tame Range CyTrONE

In this paper, we describe the architecture and implementation of the developed CyExec*. We also show the usefulness of CyExec* by comparing it to existing exercise platforms. We describe existing exercises and cyber range in Chapter 2, discuss the randomization of cyber range scenarios in Chapter 3, describe the implementation in Chapter 4, and present the comparative validation in Chapter 5.

2 INFORMATION SECURITY EDUCATION

2.1 Various Hands-on Environments

In information security education, hands-on learning is prevalent, such as CTF (Capture The Flag). Learners work on problems like a game using security knowledge or lab work to learn the attack methods and vulnerabilities through experience. Many of these are open to the public, and learners can set up their environment to work on the problems (Chothia and Novakovic, 2015).

On the other hand, in cyber range, teams are divided into roles, such as CEOs, security personnel and engineers, and they work on the exercise from their perspective. The environment in which the exercise is conducted is a reproduction of the real system environment, and the organization's incident response skills are developed through scenarios that simulate real security incidents (Maki et al., 2020).

The cyber range exercises are complex environments, difficult to set up, and are not available to the public. Some educational institutions use them in their professional courses, but companies use commercially available cyber range systems to train their professionals in many cases. Table 1 shows a comparison of hands-on exercises in information security education.

CTF and lab-work are easy to prepare the environment and comfortable for individuals to learn. The cyber range is highly effective in education but requires the preparation of appropriate scenarios and environments. Therefore, it is not easy to conduct exercises and utilize them in the educational curriculum (Chapman et al., 2014; E et al., 2017; Raj et al., 2016).

2.2 Responding to Cyber Range Issues

To address the problem of developing cyber range scenarios and building an environment, we considered the following.

- Easy to provide a large number of scenarios.
- Even if the content of the scenario is leaked, the educational effect will not be compromised.
- The environment is capable of executing many scenarios simultaneously.

Although cyber range scenarios are the cornerstone of the exercise, the educational outcome is limited if the same scenarios are always being offered. Also, if many teams are working on the same scenario simultaneously, the scenario's content could be leaked from the situation of other teams. If the exercise is conducted online, students may share tips and answers, which is not a positively affect for teaching purposes. Additionally, to run multiple scenarios, many virtual instances need to be running simultaneously, and the environment needs to be prepared, replicated, replaced, started, and closed quickly (Costa et al., 2020).

To address the above, we referred to the concept of SecGen, which is a publicly available platform that provides many exercise environments (Schreuders et al., 2017). The main concepts of SecGen are as follows.

- Provides a randomizable, flexible, and generic method for security education used in CTF and security lab exercises and simulations.
- Outputs a set of VMs, including server, client and network configurations, software and configuration vulnerabilities, and randomizes their various component to create richer scenarios.
- Design and implement a specification for generating scenarios randomly.

These concepts can be used in a cyber range environment and are essential for providing students with many organized learning opportunities. Even on the cyber range, incorporating random components into scenarios can provide many scenarios, providing learning opportunities, and improving educational effectiveness. However, some of the randomization components targeted by SecGen lack or have different use for cyber range, since it has different learning objectives and target systems. The cyber range characteristics must be considered in determineing which components should be randomized in the scenario. That will be discussed in Chapter 3.

Also, cyber range requires more virtual instances than other exercises. As a result, it requires a lot of hardware resources and high-performance equipment. Running multiple scenarios may require even more high-performance equipment, which is not practical. Therefore, we considered carrying on the concept of CyExec, a cyber range platform that leverages containerized virtualization with Docker (Li et al., 2017). Specifically, we have developed CyExec*, which implements the concept of scenario randomization in CyExec. CyExec* is envisioned as a realistic exercise system that can be integrated into educational curricula and will be discussed in detail in Chapter 4.

3 RANDOMIZE CYBER RANGE SCENARIOS

3.1 DAG-based Cyber Range Scenario

In examining the randomization of cyber range scenarios, we analyzed an example of a common cyber range scenarios as show on Figure 1.

The cyber range scenario has several milestone points within the overall scenario. The operations and actions that reach those milestones are a single scenario that deals with individual attack methods, similar to CTF and lab work. The overall scenario can

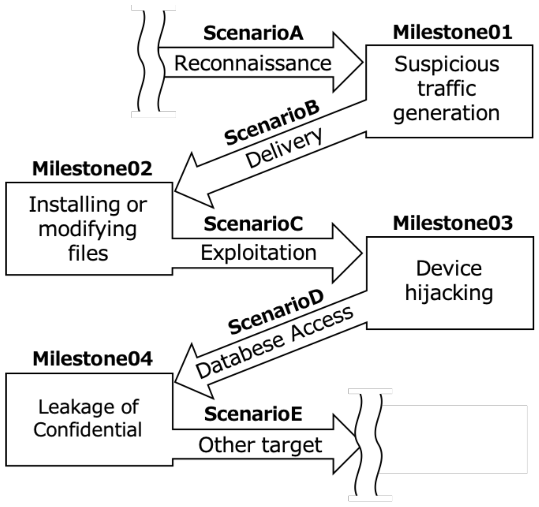


Figure 1: Example of a cyber range scenario.

be thought of as a composite scenario in which each milestone is connected through a separate scenario, ultimately forming the entire incident.

There is more than one operation or action to reach each milestone. For example, the difference between a malicious file or malware being sent via email, or downloaded via a rogue website, and the difference between whether an attacker exploits a configuration issue in the operating system, or a vulnerability in a particular software in an attempt take away administrative rights. Some of these scenarios could read to the same outcome, even if the attacker uses different means. We thought that if we could fix the milestones and incorporate a random element into the scenarios leading up to them, we could create a random cyber range scenario. Figure 2 shows an image of a cyber range scenario that incorporates randomization.

The randomized cyber range scenario takes the form of a graph, with the milestone considered to be the same state as the vertex and the scenario directed to the next milestone as the edge. Since the attack is directed towards the final target, the scenario does not consider the possibility of going back to a previous milestone or looping back to the same location. Therefore, a cyber range scenario that takes randomness into account is a directed acyclic graph (DAG).

When developing a scenario, it is important to ensure that the overall scenario is a DAG and consider multiple scenarios heading towards each milestone. This method can provide a scenario pattern for the total number of paths in the graph. Multiple random scenarios with different paths but the same objectives allow participants to experience security incidents in a variety of situations.

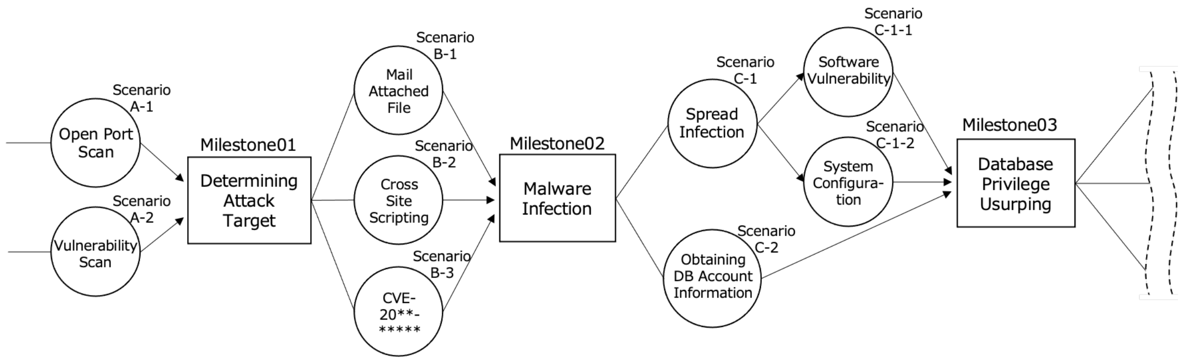


Figure 2: Example of a randomized cyber range scenario.

3.2 Consideration of Randomizable Components

In 3.1, we showed how to develop a random scenario using DAG. In order to conduct a cyber range exercise, it is necessary to build a scenario reproduction environment within the cyber range. To examine the components that can be randomized during the construction of the cyber range environment, we identified the components considered in SecGen (Schreuders et al., 2017), a VM-based scenario generation tool. Table 2 shows the results of the examination of the components that can be randomized.

The content of SecGen can be incorporated into the CyExec* environment as well. Almost any element can be considered for randomization and can be used to cyber range exercise. These contents can be randomized as long as they do not interfere with the progress of the scenario. Even if the scenario leads to the same milestone, they are provided as new situations, such as different environments and settings to operate in, or a vulnerability used for an attack.

In the case of SecGen, there is an internal network of single or multiple VMs, so there is no need to consider the external networks, as long as the necessary components work. In CyExec*, to replicate the real-world network configuration, not only do we need to replicate the vulnerabilities and configuration flaws, but we also need to consider randomization of internal/external network configurations, including not only vulnerabilities and configuration flaws, but also security devices such as IPS and firewalls, etc.

When developing scenarios, we will develop DAG-based randomized scenarios by identifying these randomizable components and branching them out with content that can lead to same milestones.

4 DEVELOPMENT AND IMPLEMENTATION

4.1 Consideration of Implementation Methods

To implement the randomized cyber range scenarios that we have studied, we investigated the implementation in CyExec, a container-based cyber range. CyExec uses Docker to implement a virtual environment. Docker is a platform for containerized virtualization and has the following features that make it useful for cyber range implementations (Raj et al., 2016; Docker, c).

- Reduced resource consumption due to virtual instance increase
- Environment suitable for frequent changes and disposables
- Managing image and network configuration with Dockerfile and Docker-compose

Docker behaves like an independent machine by grouping and isolating only the processes that are necessary to run the virtual environment. Because other resources are shared with the host, the Overhead is low, and the operation is fast and lightweight. Since it does not occupy memory and CPU like VMs, the load is low even when many virtual instances are launched in the cyber range.

Containers have a smaller image size than VMs, because containers require a minimal configuration. In addition, the Union File System (UFS) allows frequent changes and partial additions to the base image to be made efficiently. It is often used in verification environments that do not require data persistence, and is suitable for disposable environments such as exercises. The images can all be described in a simple text file called Dockerfile (Docker, a). In the case of

Table 2: Scenario Components.

Random Components	SecGen	CyExec*
OS (virtual instance)	Single OS with multiple services	Multiple OS and services can be added flexibly
Network	Internal network of VMs and hosts	Internal/External complex multiple networks
Network Services	such as FTP, IRC, HTTP, NFS Internal Network Services	Including IPS/IDS and monitoring services. Internal/External services
System Configuration	Configuration within a single OS and the Installed Software	Multiple OS and services, Softwares configuration
Vulnerabilities	Independent Vulnerabilities Configuration/Softoware	Equivalent vulnerability resulting in equivalent milestones
CTF-style challenges	Addressing the above issues or attacking vulnerabilities	Advanced attacks such as targeted attacks and incident response

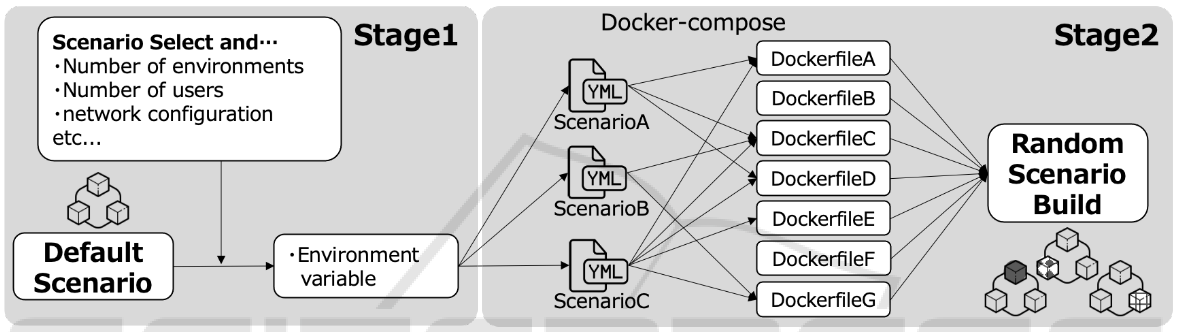


Figure 3: Image of CyExec* in action.

SecGen, we use many platforms such as Ruby, Vagrant, Puppet, Virtual-Box, to build VMs. Although these environments are generally accessible, they may not work correctly due to various factors, such as differences in versions and modules' interaction. Using Docker as the primary platform makes it easier to implement containers with the necessary configurations instead of VMs.

CyExec* uses many Dockerfiles to build containers of various configurations and prepare multiple environments. To build a network environment using multiple Dockerfiles, we use Docker-compose to automate (Docker, b). Docker-compose can start an environment with multiple container images, including network configuration. By providing multiple Docker-compose files, it enables randomization or changing the environment by specifying the Dockerfile to be used on startup.

Figure 3 shows an image of CyExec* in action.

CyExec*, like SecGen, consists of two stages. Stage 1 is based on the default scenario and defines, the number of environments to generate, the number of students to connect to (and the devices to prepare for), etc. Based on this, CyExec passes the scenarios and the necessary arguments to Docker-compose, specifies the Dockerfile needed to execute the scenarios, builds the container, and launches the exercise environment at stage 2.

4.2 Implementing the Default Scenario

CyExec* provides a default scenario with a base system configuration. Additional randomness can be accommodated by modifying some of the default scenarios. The default scenario environment can also be used as a place to learn attack and defense techniques or to develop scenarios.

Default scenario is automatically generated by Dockerfile and Docker-compose, which is highly portable, and any environment running Docker can build an equivalent environment. There is no need to develop complex programs or modules for scenario development. Scenario development can be handled by adding a Dockerfile and editing Docker-compose.yml file. Figure 4 shows the configuration of the default scenario.

In the exercise, students requires a device that can be operated in the cyber range. Also, this device will also include a GUI desktop environment that allows connection from the host. In CTFs and lab work, the host on which the virtual environment is built is also part of the exercise environment. Because the cyber

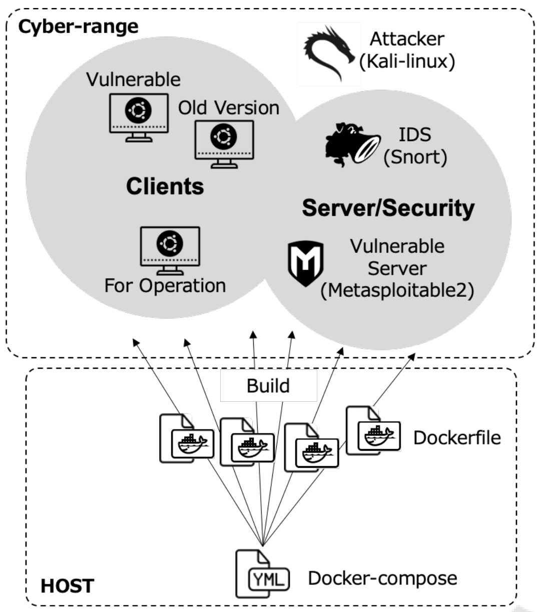


Figure 4: Configure the default scenario.

range reproduce a real-world network, providing an operational devices within the exercise environment can make the exercises more realistic.

Also, as a vulnerable environment, Metasploitable2 is used to handle a variety of attack scenarios coming from the web and vulnerable applications. Metasploitable2 is a server that deliberately holds vulnerable configurations and vulnerable versions of software, and is used for penetration testing and other purposes (Rapid7,). Officially, Metasploitable2 is offered in VM, but we implemented it by containerizing. We implemented snort as an IDS that can detect attacks from kali-linux (Security, ; team,), which is prepared as an attacker. Other vulnerabilities with different settings and versions were also implemented on multiple client devices.

All virtual instances are implemented in containers, but are as usable as VMs, and even attack scenario can be executed at startup by writing into a Dockerfile or making a script. By using this default scenario, the exercises can be run from attack to defense. However, this default scenario is not prepared for a specific attack or vulnerability in mind. The idea is to add new equipment and service containers to this environment and expand the types of randomness by changing the attack methods and forensic content to provide more scenarios.

4.3 Adding Randomness

To test the addition of randomness, we considered adding a random scenario with multiple attack methods to the default scenario environment. Vulnerable application running on Metasploitable2 can be attacked using the exploit module in kali-linux. If this attack's result creates the same state and leads to the next scenario, this attack's content can be chosen at random. Table 3 shows the vulnerable applications running on Metasploitable2 and the equivalent exploit modules that leads to the same milestone.

Table 3: Vulnerable Applications and Equivalent Exploit Modules.

Vulnerable Applications	exploit on msfconsole
vsftpd	exploit/unix/ftp/ vsftpd_234_backdoor
php	exploit/multi/http/ php_arg_injection
Samba	exploit/multi/samba/ usermap_script
PostgreSQL	exploit/linux/postgres/ postgres_payload
UnrealIRCD	exploit/unix/irc/ unreal_ircd_3281_backdoor
distccd	exploit/unix/misc/distcc_exec
Ruby DRb RMI	exploit/linux/misc/ drb_remote_codeexec

As a result of attacks using these exploit modules, arbitrary commands can be executed remotely on Metasploitable2. Since they can execute commands with root privileges, they can perform various actions, such as obtaining information or tampering with files, which leads to the next attack scenario. This way, the scenarios can be swapped and selected randomly if a similar situation is created, using a different vulnerability or attack technique. Because same circumstances can be created even from different exploit modules, you can swap them around and have it choose randomly. Therefore, you can add randomness to the scenarios.

CyExec* executes attack scenarios by writing them in a Dockerfile or loading a script. In other words, selecting which Dockerfile to use determines the scenario. To make randomization easier, we placed the Dockerfile in a separate directory for each scenario; selecting a directory in the Docker-compose launches the scenario based on where the Dockerfile is placed. Figure 5 shows an example of the directory structure.

Place the Docker-compose.yml file in the top directory. The environment will be built using the Dockerfile in a subdirectory. Arrange the branches of the scenario as symbolic links so that the directory structure takes a DAG form. The scenario struc-

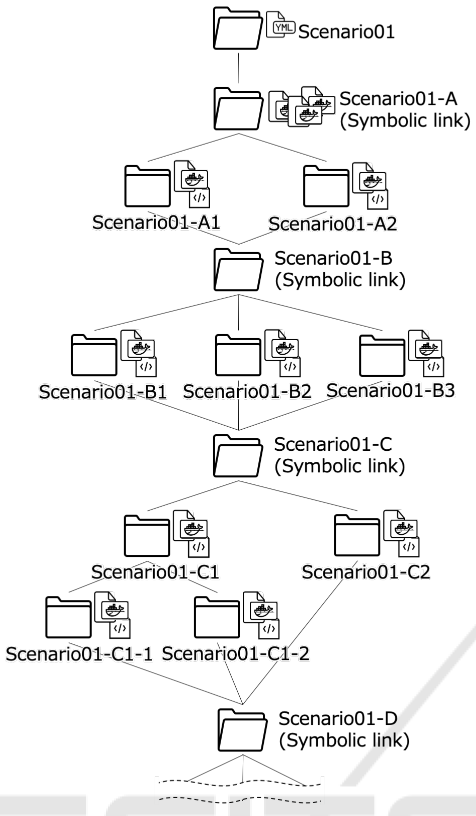


Figure 5: Example of CyExec* directory structure.

ture matches the directory structure. The Docker-compose.yml file specifies the directory where the Dockerfile is located. By making this a variable, it made it possible to choose any scenario route or randomize.

5 VERIFICATION AND COMPARISON

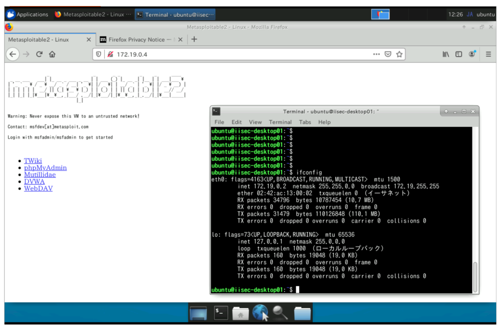
5.1 Verification of Operation

To verify the operation of CyExec*, we ran the default scenario. For reference, here is the example Dockerfile of the client device.

```
FROM ubuntu:18.04
LABEL maintainer="Ryotaro Nakata"

ARG host_name="cyexec*_client01"
ENV DEBIAN_FRONTEND=noninteractive \
    HOSTNAME=$host_name

#Locale and Language setting
RUN apt-get update && \
    apt-get install -y ibus-mozc

#User setting
ARG root_password="root"
ARG user_name="ubuntu"
ARG user_password="password"
RUN echo root:$root_password | \
    chpasswd && apt-get update && \
    apt-get install -y openssl sudo \
    && useradd -m -G sudo $user_name -p \
    $(openssl passwd -1 $user_password) \
    --shell /bin/bash

#RDP setting
RUN apt-get update && \
    apt-get install -y xfce4 xfce4-terminal \
    xfce4-goodies xrdp && adduser xrdp \
    ssl-cert && update-alternatives --set \
    x-terminal-emulator \
    /usr/bin/xfce4-terminal.wrapper
COPY ./config/xrdp/sesman.ini \
     /etc/xrdp/sesman.ini
COPY ./config/xrdp/xrdp.ini \
     /etc/xrdp/xrdp.ini
COPY ./config/xrdp/default.pa \
     /etc/xrdp/pulse/default.pa
EXPOSE 3389

#Startup setting
RUN apt-get update && \
    apt-get install -y supervisor
ADD ./config/supervisord/* \
    /etc/supervisor/conf.d/

#Install Preferred package
RUN apt-get update && \
    apt-get install -y \
    git tig gedit nano \
    wget curl net-tools firefox\
    build-essential \
    software-properties-common

#Install Network tools
RUN apt-get update && \
    apt-get install -y \
    iputils-ping inetutils-traceroute \
    net-tools

# Clean up
RUN apt-get clean && apt-get autoremove \
    && rm -rf /var/cache/apt/archives/* \
    /var/lib/apt/lists/*

CMD ["bash", "-c", "/usr/bin/supervisord \
    -c /etc/supervisor/supervisord.conf"]
```

This client device is available in all scenarios, as a device for participants to study and review incidents. We also built Kali-linux, Metasploitable, and Snort using container images published on DockerHUB. These containers can be started and run in scenarios based on the Dockerfile description. Figure 6 shows a screenshot of the container connection and operation.

We have confirmed that everything works fine. All containers are working correctly, including access to the Metasploitable2 web screen by launching a browser from the operating terminal's GUI desktop. This state alone is sufficient enough to be used as a simulation environment. Students will carry out the exercise by performing various checks and examinations from this client device.


Applications
Metasploitable2 - Linux
Terminal - ubuntu@iisec-desktop01: ~
172.19.0.4
File Edit View Terminal Tabs Help

```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

```
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$
ubuntu@iisec-desktop01:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.0.2  netmask 255.255.0.0  broadcast 172.19.255.255
        ether 02:42:ac:13:00:02  txqueuelen 0  (イーサネット)
        RX packets 34796  bytes 10787454 (10.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31479  bytes 110126848 (110.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (ローカルループバック)
        RX packets 160  bytes 19048 (19.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 19048 (19.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@iisec-desktop01:~$
```

Figure 6: Desktop operations in a launched container.

5.2 Running Randomize Scenario

We added an exploit that sends malware to the default scenario to verify that it can run a random scenario in CyExec shown by Figure 7 .

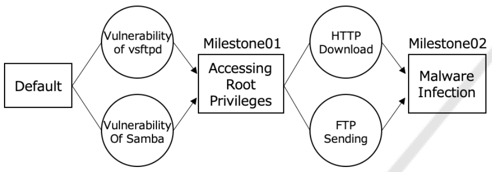


Figure 7: Randomization.

With the randomization scenario shown in Figure 7, we confirmed that all patterns of the scenarios could be executed, as scenarios by changing the exploit type and how to send malware. Here's an example of the Docker-compose.yml file used to select the scenarios

```
version: '3'

services:
  client01:
    build:
      context: .
      dockerfile: >
      -./scenario01-A/Dockerfile_client01
  client02:
    build:
      context: .
      dockerfile: >
      -./scenario01-A/Dockerfile_client02
  vulnerable_server:
build:
      context: .
      Dockerfiles: >
      -./scenario01-A
      -/Dockerfile_metasploitable2
  IDS:
    Build: .
      context: .
      Dockerfiles: >
      -./scenario01-A/Dockerfile_snort

  scenario_A:
build:
      context: .
      Dockerfiles: >
      -${SCENARIO_01_A:-"./scenario01_A1"}
  scenario_B:
build:
      context: .
      Dockerfiles: >
      -${SCENARIO_01_B:-"./scenario01_B1"}
```

Clients and servers, which are common to the entire scenario are started at the first step. The part related to the selection of the scenario is specified by variables beforehand, but is started later so it does not interfere with the progress. In the default scenario, all the vulnerable applications shown in Table 3 are running, but some may not be necessary depending on the selected scenario route. Since they do not interfere with the scenario's execution, they are left running for this verification. However, there are times when stopping them is desirable, such as when you want to simplify the exercise or focus on performance.

In this case, we confirmed that 4 scenarios with 2x2 path can work without any problems. In the Docker-compose.yml file, you can also specify additional settings, such as network configuration, in variables. By running several scenarios at the same time and separating the network configuration, different scenarios can be used in team-based exercises.

5.3 Performance Comparison with SecGen

To verify the performance of CyExec*, we compared it to SecGen. CyExec* and SecGen have very different configurations. The default scenario of CyExec* contains 6 containers and utilizes a large image such as kali-linux, whereas the default scenario for SecGen is 1 VM image, including web server, etc. CyExec* is the more versatile environment, but it covers the content of the default SecGen scenario and can replicate an equivalent environment. Both environments are fundamental to the exercise and comparing the two can provide useful information on preparation time and hardware resource consumption.

Figure 8 shows a comparison of startup speed and storage consumption for CyExec* and SecGen, when the default scenario is launched multiple times and when the randomization of scenarios are in place.

Both CyExec* and SecGen require the necessary images to be downloaded when they are run for the first time. In particular, CyExec* uses a large number of container images and requires more download time and storage space when it is first started. However, the build time and storage consumption are less than 1/10 compared to SecGen after the second time. This is most likely the result of the efficient use of storage with UFS(Union File System) and the elimination of the startup process required for VMs , by using Docker.

Even with the addition of random scenarios, CyExec had a bit of increase in time for building changes, but was still very fast and storage intensive compared to the VM-based SecGen. Also assuming

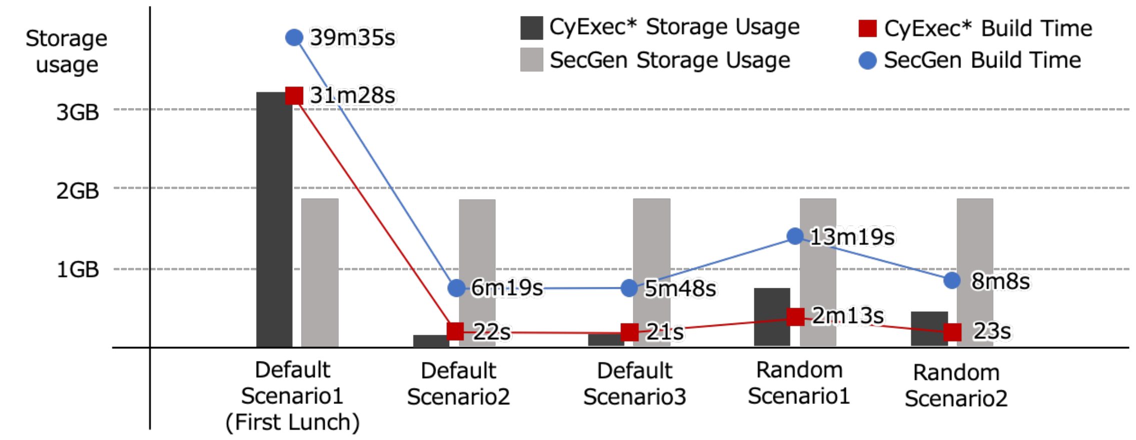


Figure 8: Comparison of startup time and storage consumption when launching several scenarios, including randomization.

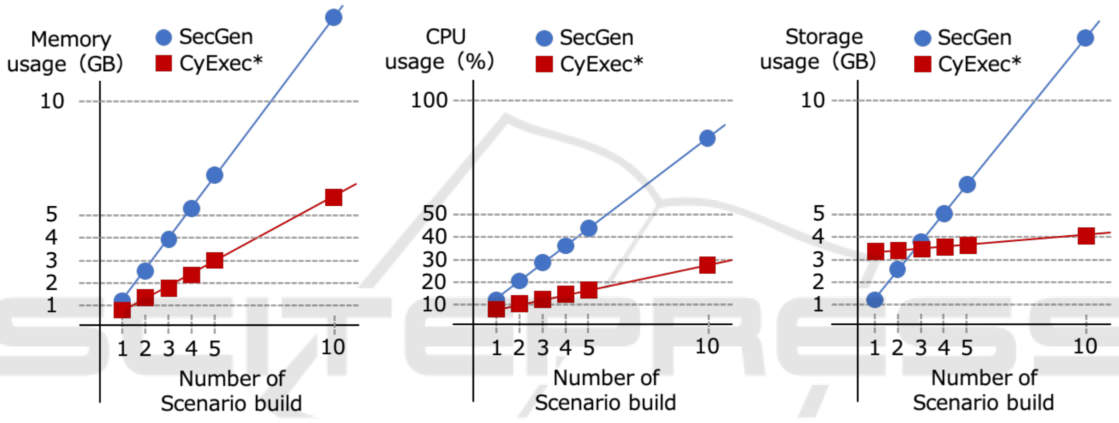


Figure 9: Comparison of resource consumption for multiple launches of the same scenario.

an exercise with many teams, we compared the consumption of hardware resources when multiple default scenarios were launched. The results are shown in Figure 9.

When built under the same hardware, the increase in hardware resource consumption was proportional to the number of environments; CyExec's default scenario had less than one-half the memory consumption and less than one-third the CPU consumption compared to SecGen's default scenario. With CyExec*, the initial storage consumption was high due to the large number of images used during the initial build. However, because consumption hardly increases even when there are multiple environments, as the number of environments increases, the consumption can be kept much lower than with SecGen. These results show that CyExec* has an advantage with an increasing number of environments, which can be very useful, especially when randomization requires multiple exercise environments.

6 APPLICATION TO SECURITY EDUCATION

The CyExec, on which this development was based on, has been used in several educational institutions and training for workers. We also consider to carry out an exercises using the randomized scenarios developed in CyExec*.

We planned to conduct the exercise at CyExec* as well to test the educational effectiveness of the exercise. However, due to COVID-19, we could not gather students to conduct the exercise.

CyExec* takes into account the effectiveness of conducting an exercise by assembling in a classroom or other settings. In addition, we expect the demand for online cyber range exercises to increase. We think important to have randomness in the scenarios, even when the exercises are conducted online.

We plan to use cloud services to apply CyExec* to the online environment, and test the effectiveness of the exercise.

CyExec* is a cyber range that can significantly reduce resource consumption and is suitable for cloud services. Randomization of scenarios can further increase its effectiveness. We have already started testing it on several cloud services, and our goal is to conduct online exercises and verify the effectiveness of its educational outcome. We will continue our research and development, and expand our information security education in the future.

7 CONCLUSIONS

The cyber range used in information security exercises is a system that allows students to learn knowledge and skills efficiently, through a highly realistic security incident experience reproduced in virtual space. However, the system is not easy to implement nor operate. In particular, scenario development requires specialized knowledge, and using the same scenario or sharing and leaking of scenario information has been a problem.

We developed the DAG-based scenario randomization technique named CyExec* on docker-based cyber range system. Multiple scenarios with the same learning objectives can enhance teaching effectiveness and prevent cheating. CyExec* makes the best use of Docker's performance, which allows us to build exercise environment efficiently, eliminating the concern of the increasing load on the system caused by the scenario number increase. In comparison to SecGen, CyExec* showed advantages of 1/3 memories, 1/4 CPU loads, and over 1/10 storage usage. This shows that it has at least three times more capability to reproduce scenarios than VM-based environments, allowing more complex environments to be run simultaneously.

Our future work includes several main directions:(i) enriching the random scenarios that can be provided by CyExec*, (ii) applying to online exercise, and (iii) testing their educational effectiveness.

ACKNOWLEDGEMENTS

This work is supported in part by the Telecommunication Advancement Foundation.

REFERENCES

Beuran, R., Inoue, T., Tan, Y., and Shinoda, Y. (2019). Realistic cybersecurity training via scenarioprogression management. In *2019 IEEE European Symposium on Security and Privacy Workshops*.

Chapman, P., Burket, J., and Brumley, D. (2014). Picoctf: A game-based computer security competition for high school students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA. USENIX Association.

Chothia, T. and Novakovic, C. (2015). An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*.

Costa, G., Russo, E., and Armando, A. (2020). Automating the generation of cyber range virtual scenarios with vsdl.

Docker. Dockerfile reference. https://docs.docker.com/engine/reference/builder/.

Docker. Overview of docker compose. https://docs.docker.com/compose/.

Docker. What is a container? a standardized unit of software. https://www.docker.com/resources/what-container.

E, I. C., F, T. M., and Jean, K. (2017). Labtainers: a framework for parameterized cybersecurity labs using containers.

Li, Z., Kihl, M., Lu, Q., and Andersson, J. A. (2017). Performance overhead comparison between hypervisor and container based virtualization. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 955–962.

Maki, N., Nakata, R., Toyoda, S., Kasai, Y., Shin, S., and Seto, Y. (2020). An effective cybersecurity exercises platform cyexec and its training contents. In *2020 International Conference on Advances in Education and Information Technology(AEIT'20)*.

Raj, A. S., Alangot, B., Prabhu, S., and Achuthan, K. (2016). Scalable and lightweight ctf infrastructures using application containers. In *2016 USENIX Workshop on Advances in Security Education (ASE 16)*.

Rapid7. Metasploitable2. https://docs.rapid7.com/metasploit/metasploitable-2/.

Razvan, B., Cuong, P., Dat, T., Ken-ichi, C., Yasuo, T., and Yoichi, S. (2017). Cytrone: An integrated cybersecurity training framework. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017): 157-166*.

Schreuders, Z. C., Butterfield, E., and Staniforth, P. (2015). An open cloud-based virtual lab environment for computer security education. In *The first UK Workshop on Cybersecurity raining & EducationVibrant Workshop 2015*.

Schreuders, Z. C., Shaw, T., Shan-A-Khuda, M., Ravichandran, G., and Keighley, J. (2017). Security scenario generator (secgen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting ctf events. In *2017 USENIX Workshop on Advances in Security Education(ASE'17)*.

Security, O. Kali linux — penetration testing and ethical hacking linux distribution. https://www.kali.org.

team, S. Snort network intrusion detection & prevention system. https://www.snort.org/.

Vincent E Urias, William M.S Stout Brian Van Leeuwen, H. L. (2018). Cyber range infrastructure limitations and needs of tomorrow.