PAPER

LiNeS Cloud: A Web-Based Hands-On System for Network Security Classes with Intuitive and Seamless Operability and Light-Weight Responsiveness*

Yuichiro TATEIWA[†a)], ***Senior Member***

SUMMARY We consider network security exercises where students construct virtual networks with User-mode Linux (UML) virtual machines and then execute attack and defense activities on these networks. In an older version of the exercise system, the students accessed the desktop screens of the remote servers running UMLs with Windows applications and then built networks by executing UML commands. However, performing the exercises remotely (e.g., due to the COVID-19 pandemic) resulted in difficulties due to factors such as the dependency of the work environment on specific operating systems, narrow-band networks, as well as issues in providing support for configuring UMLs. In this paper, a novel web-based hands-on system with intuitive and seamless operability and lightweight responsiveness is proposed in order to allow performing the considered exercises while avoiding the mentioned shortcomings. The system provides web pages for editing device layouts and cable connections by mouse operations intuitively, web pages connecting to UML terminals, and web pages for operating X clients running on UMLs. We carried out experiments for evaluating the proposed system on the usability, system performance, and quality of experience. The subjects offered positive assessments on the operability and no negative assessments on the responsiveness. As for command inputs in terminals, the response time was shorter and the traffic was much smaller in comparison with the older system. Furthermore, the exercises using nano required at least 16 kbps bandwidth and ones using wireshark required at least 2048 kbps bandwidth.

key words: *network security, e-learning, virtual machine*

1. Introduction

The network security exercise that we considered consist of two phases bellow. In both phases, the students repeat constructing networks, executing tools, observing networks, and reconfiguring networks.

1. The students confirm the roles of network devices and the effects of tools for attack, diagnostic and so on by experiments. For example, they build vulnerable networks as experimental environments, and then attack the networks by executing attack tools. Next, they detect the attacks by analyzing system logs with monitoring tools. And finally, they place firewalls to the networks in order to defend them from attacks.
2. The students consider the relationship between the service provision and use, the attacks, and the defenses. They devise stories in which the network administrators and crackers appear, and afterwards, successful attacks and defenses occur, followed by the need to experiment with the attacks or defenses in their networks. The following example of the stories consists of two characters and two attacks. A stupid administrator adds a server with initial password to a network. After a period of hours, a cracker successfully intrudes the server by password cracking, and then he/she attacks another server from the first server by Dos attack.

User-mode Linux [1] (UML) is a program that runs on Linux and virtualizes a Linux computer and within it, uml_switch is a tool that works like a switching hub. By connecting the network interfaces of UMLs to a uml_switch, a virtual network can be realized on their host computer. Since UML has been developed for a long time (we confirmed a version published in 2001), it can be used for network security exercises using security holes that have been discovered over a long period of time. Furthermore, since UML is also free to use, it is easy to be adopted in schools.

In the older version of the exercise system, the students started vSphere Client [2] on Windows personal computers (PCs) in a laboratory and operated the desktop screen of Debian Linux 3.1 running on remote servers with the application. Figure 1 shows a virtual network that connects a client and a server to a switching hub, where the windows of the client and server are used as their own terminals. The virtual devices are created and connected by executing UML commands on the Debian Linux terminals.

However, this exercise environment has some shortcomings, as follows.

1. During the COVID-19 pandemic, the importance of having all participants, students, and instructors in the same room or facility has dramatically decreased. In particular, this exercises should be carried on while the students are at home and not in a lab. However, this seems to be difficult as setting up an effective work environment at home raises challenges such as the ones listed below.
    - The operating systems (OSs) that the students have access to through their PCs and tablets may not be Windows, so some of them might not be able to use the application vSphere Client.

Manuscript received November 24, 2021.
Manuscript revised March 25, 2022.
Manuscript publicized June 8, 2022.
[†]The author is with Nagoya Institute of Technology, Nagoya-shi, 466–8555 Japan.
[*]This is a paper on system development.
a) E-mail: tateiwa@nitech.ac.jp
DOI: 10.1587/transinf.2021EDK0006

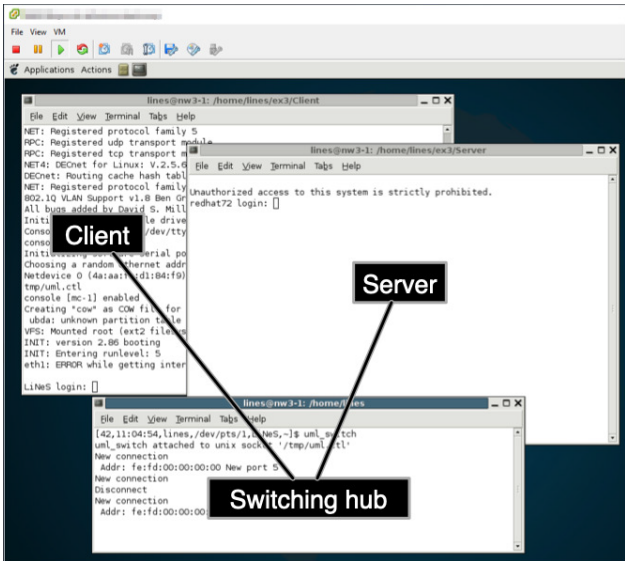


Fig. 1 Older version of the exercise system.

- Some students might have internet access via narrow-band networks that may cause difficulties in operating the desktop screens generated by screen transfer.
- Poor communication with students by phone or e-mail could cause difficulties for instructors and assistants in supporting them during the construction of the exercise environment.

2. Causes such as the following ones could prevent the students from working on the exercises efficiently.
   - Since it is difficult to distinguish between terminals, they may mistakenly operate a host terminal or another UML terminal instead of the desired UML terminal.
   - The network topologies of virtual networks are implemented by executing UML commands. The students may fail to build their virtual networks quickly and accurately because of difficulties in designing the command arguments and even due to typos.
   - The students may fail to remember the network topologies correctly, which could result in unintended tasks. For example, the students find how to defense attacks with firewalls through trial and errors. In this work, they place/remove firewalls to/from various places in their networks. Some students might doubly place firewalls to the similar places.
3. Since security holes are plugged as they are discovered, older OSs and tools may be required for the exercises, depending on the attacks we want the students to practice. Moreover, newer monitoring tools tend to be equipped with more features and easier to use UIs. Additionally, it is desirable for the students to select tools used for experiments in the second phase of the considered exercise from the whole tools of the first phase. For these reasons, we would like to build virtual networks consisting of various versions of UML. However, the following reasons might prevent this.
   - There are cases that the OS running UML (hereinafter called the host OS) is determined by the UML version: we confirmed that some newer host OSs were not able to run older UMLs and vice versa. Therefore, it is difficult to build virtual networks consisting of various versions of UMLs in a single host OS.
   - UMLs and uml_switches are not equipped with functions enabling exclusive connections between them while running on different host computers.

We proposed a web-based hands-on system **LiNeS Cloud** using UML with intuitive and seamless operations and light-weight responsiveness [3]. This novel system has the following features.

- It builds virtual networks on remote servers and provides dynamic web pages for operating the networks. In addition, both text transfer and screen transfer are differently employed for communication between the user interfaces (UIs) and the servers. Consequently, the students can use the exercise environment comfortably and easily through browsers (this solves problem (1)).
- It provides dynamic web pages where the students can edit the topologies of the virtual networks by mouse operations. Furthermore, UML terminals are mapped to the networks drawn on the web pages. They enable the students to intuitively form virtual networks (this solves problem (2)).
- It builds overlay networks implemented by tunnels that can forward Ethernet frames between virtual devices running on various remote servers. This enables the implementation of virtual networks that contain various versions of UMLs (this solves problem (3)).

We proposed the design and implementation of the system in [3], described its use in our classes in [4], and discussed the response time, communication traffic, and user questionnaire of the system in [5]. In this paper, we discuss the quality of experience (QoE) in responsiveness for the network speed between the UIs and the remote servers, in addition to the contents in [3] and [5].

2. Related Work

NetPowerLab [6] provides virtual networks to learners, including virtual devices implemented by UMLs on remote servers. The learners can edit the network topologies intuitively by mouse operations, and they can use the terminals of the virtual devices through web UIs. However, NetPowerLab is not suitable for the exercises considered in this work because the learners cannot use X clients in the virtual devices and cannot build virtual networks composed of various versions of UMLs.

V-Lab [7], CloudLab [8], and ThoTh Lab [9] provide

learners with virtual networks on remote servers. Concerning virtual devices, Xen is used in V-Lab, Xen and Docker in CloudLab, and QEmu in ThoTh Lab. First, the learners define network topologies and virtual device configuration on web pages and then register these into the systems. Then, the systems build virtual networks based on the provided definitions. Finally, the learners select virtual networks and operate the virtual devices via web pages in V-Lab and ThoTh Lab, and via SSH clients and X servers, prepared and configured by themselves, in CloudLab. However, the learners cannot efficiently repeat processes of changing the topologies and operating the devices because the processes are divided into three phases: defining network topologies and configuration of the virtual devices, creating virtual networks based on the definitions, and operating the virtual devices. For this reason, they are not suitable for the considered exercises, which can require the students to change their network topologies as we mentioned in the first paragraph of Sect. 1.

CyTrONE [10] provides a cyber range training environment whose network devices are implemented using Kernel Virtual Machine (KVM) [11] and Amazon Web Services (AWS) [12]. Before exercises, instructors input the definition of virtual networks and security incidents into CyTrONE. In exercises, students access to virtual devices via SSH or VNC clients. Furthermore, the environment visualizes network topologies and basic states of devices on web browsers. However, CyTrONE is not suitable for the exercises considered in this work because the environment is not equipped with the functions that students edit network topologies seamlessly in exercise.

3. Basic Technology

3.1 User-Mode Linux

UML is a program that runs on Linux and virtualizes a Linux computer. When a UML kernel program (a Linux binary file) is executed with the file path of a root file system, the corresponding UML instance is created. Even non-root users in the host OS can get administrative privileges in UML instances and are allowed to install software and provide services. In addition, new devices can be attached without duplicating root file systems (i.e., immediately) because the changes from the original root file systems are saved into other files.

3.2 TAP and Bridge

TAP is a virtual network interface in Linux that can handle Ethernet frames. UML internal applications can send and receive Ethernet frames to and from the outside of the UML through the TAP instance that is generated on the host OS and connected to a UML internal network interface (e.g., eth0).

Bridge is a virtual network interface in Linux that forwards Ethernet frames like a network switch. Multiple TAPs can be connected to a single Bridge. By connecting TAPs connected to UMLs to a Bridge, a virtual network can be formed as if the UMLs were connected to a network switch. The network settings in each UML do not interfere with network settings in the host OS and other UMLs.

3.3 Tunneling

Some tunneling technologies can transfer Ethernet frames between devices connected with tunnels (hereinafter referred to as Ethernet tunnels). When connecting a device to another one with a new tunnel, a virtual network interface (hereinafter referred to as a tunnel interface) is created in each device. Ethernet frames are exchanged through the tunnel interfaces. Therefore, by connecting a TAP or Bridge in a host to a tunnel interface, Ethernet frames can be forwarded to the TAP or Bridge connected to the tunnel interface in another host, thus enabling the exchange of Ethernet frames between UMLs and Bridges in different hosts.

4. Exercise Overview

In the considered exercises, students are requested to solve problems in the textbooks for achieving the objectives stated in Table 1. For example, the students exercise defenses

Table 1 Summary of the network security exercises

Chapter	Tasks	Objectives
Network construction	• Build networks with servers, clients, switching hubs, routers, and firewalls	• Can design settings for basic services and apply them to servers • Can explain the concept of application protocols • Can analyze system logs in ordinary use
Attacks	• Attack networks by executing tools (DoS/DDoS, password cracking, packet sniffing, backdoors, buffer overflow, session hijacking, root hacking) • Observe the state and logs of the victim machines and networks	• Can explain the mechanisms of the attacks • Can detect the attacks by analyzing the state and logs of machines and networks
Defense	• Filter packets with iptables	• Can explain the mechanisms of firewalls • Can design appropriate rule-sets for satisfying the requirements • Can set up iptables
Cracker programming	• Create vulnerable programs • Create cracking programs for the vulnerability • Create patch programs for the vulnerability	• Can design secure programs • Can find vulnerable codes in programs • Can fix the codes

against DoS attacks with firewalls, as follows.

1. They construct the networks specified in the textbooks. The textbooks define the network topology (A client and a server are connected to a switching hub) and define settings of the devices.
2. They make sure that users of the clients can use the web services on the servers with telnet clients.
3. They add the devices for attacks (hereinafter referred to as black hats) to the networks and execute DoS attacks from the devices to the servers.
4. They make sure that users of the clients cannot use the web services on the servers with telnet clients.
5. They make sure that the servers are suffering DoS attacks by executing the netstat command on the servers.
6. They add firewalls to the networks and set the firewalls so as to filter packets from the black hats.
7. They make sure that users of the clients can use the web services on the servers with telnet clients.

5. System Design

5.1 Overview

We used UMLs to implement servers, clients, routers, and firewalls and Bridges to implement switching hubs.

In the system overview presented in Fig. 2, the simulation servers accept remote operations from the configuration server and implement virtual networks by employing virtual devices. The configuration server provides web pages for editing the connections among virtual devices (hereinafter referred to as topology pages), web pages for input/output with UML terminals (hereinafter referred to as terminal pages), and web pages for input/output with UML X clients (hereinafter referred to as X pages). The relay server enables the exchange of Ethernet frames between the tunnel interfaces of two simulation servers by connecting the tunnel interfaces of the Ethernet tunnels connecting themselves to the simulation servers using Bridges. The terminal server relays the input and output between the terminal pages and the UML terminals. The X forwarding server accepts drawing requests from UML X clients and draws the output on the X pages. The student can manage virtual networks with

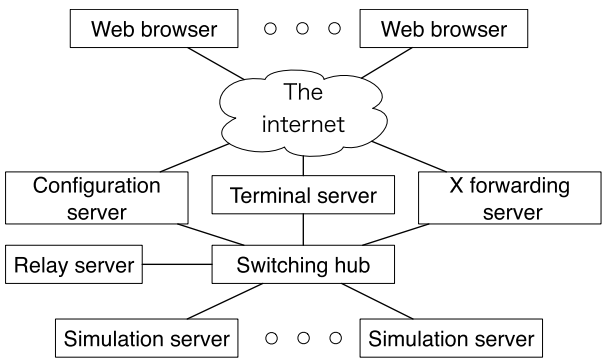


Fig. 2 System overview.

Windows, Mac, Android, and IOS web browsers.

While the host OS for UML depends on the UML version, instructors might want to adopt older versions of UML for exercises. This means that UMLs may be run on OSs with security holes due to expiration of support. Exposing severs with such OSs to the Internet is dangerous on system operation. On the other hand, a newer OS is more secure and efficient for developing and providing Web UIs. Therefore, in the considered system design, we distinguish UML host computers (i.e., simulation servers) from other server computers and block the simulation servers from accepting connections from the Internet.

5.2 Topology Page

Topology pages display network topologies, some settings and states of virtual devices, and accept user input for them. To improve their responsiveness, the pages cache a part of data and make responses against a part of user input based on the cache (i.e., such input is not processed in the configuration server). Some examples of such responses are the verification for the combination of connectable devices and the calculation for drawing data updated by moving virtual device icons (the latter is reflected in the server at regular intervals).

The pages draw network topologies with svg elements of html5. Document Object Model enables users to edit the drawings by mouse operations. In the system, the students place, remove, move, boot, and halt devices and connect and disconnect cables between devices by mouse operations.

The pages achieve asynchronous request/response communications to the configuration server without page transitions using XMLHttpRequests. They send user operations as requests to the configuration server and then receive the processing results as responses. For example, when a student moves a device on a topology page, the browser sends its request to the configuration server and accepts the input from the student without waiting for responses. This implementation enables the students to move to other editing tasks which are not affected by the result of moving the device.

5.3 Configuration Server

The configuration server receives requests sent from topology pages, processes them as shown in Table 2, and then returns responses. The ledger in the table is managed by this server, and the operations in the relay server and the simulation servers are remotely executed.

The runnable kernel version of UMLs depends on the OS of the simulation servers. In addition, due to the resource limitation of the server computers, a single computer has no capacity for running all UMLs for the students at the same time. Therefore, the configuration server assigns the UMLs to different simulation servers based on the types of devices used by the students. Virtual networks consisting of UMLs and Bridges running on a simulation server are

formed by connecting UML TAPs to Bridges. Virtual networks of UMLs and Bridges running on different servers are formed by connecting UML TAPs to Bridges with Ethernet tunnels.

6. Prototype System

The configuration server is implemented with node.js [13], the topology pages are implemented with javascript, jquery [17], ajax, SVG.js [15], the terminal pages are im-

Table 2 Request processing in simulation servers.

Request	Processes
Assigning a UML	The server registers the UML in the ledger, generates a unique TAP name and the TAP instance in a simulation server.
Assigning a switch	The server registers the switch in the ledger, generates a unique Bridge name and the Bridge instance in a simulation server.
Connecting a cable	The server registers the cable in the ledger and processes based on the simulation servers running the devices connected by the cable. If the devices run on a single simulation server, the configuration server connects the UML TAP to the Bridge. Otherwise, it constructs an Ethernet tunnel from each simulation server to the relay server, and it either connects the tunnel interface and the UML TAP to a Bridge or connects the tunnel interface to a Bridge, and then it connects the tunnel interfaces to a Bridge on the relay server.
Removing a UML	The server deletes the UML TAP on the simulation server and unregisters the UML from the ledger.
Removing a switch	The server deletes the Bridge on the simulation server and unregisters the switch from the ledger.
Disconnecting a cable	If the devices connected to the cable are running on a single simulation server, the configuration server deletes the UML TAP from the Bridge. Otherwise, the configuration server deletes either the tunnel interface and the UML TAP from the Bridge, or the tunnel interface from the Bridge on each simulation server, deletes the tunnel interfaces from the Bridge on the relay server, and then deletes the cable from the ledger.
Moving a device	The server updates the ledger.
Booting a device	The server executes the commands, either to start the UML instance or to set the Bridge instance to UP state in the simulation server, and updates the ledger.
Halting a device	The server executes the commands either to terminate the UML instance or to set the Bridge instance to DOWN state in the simulation server, and updates the ledger.

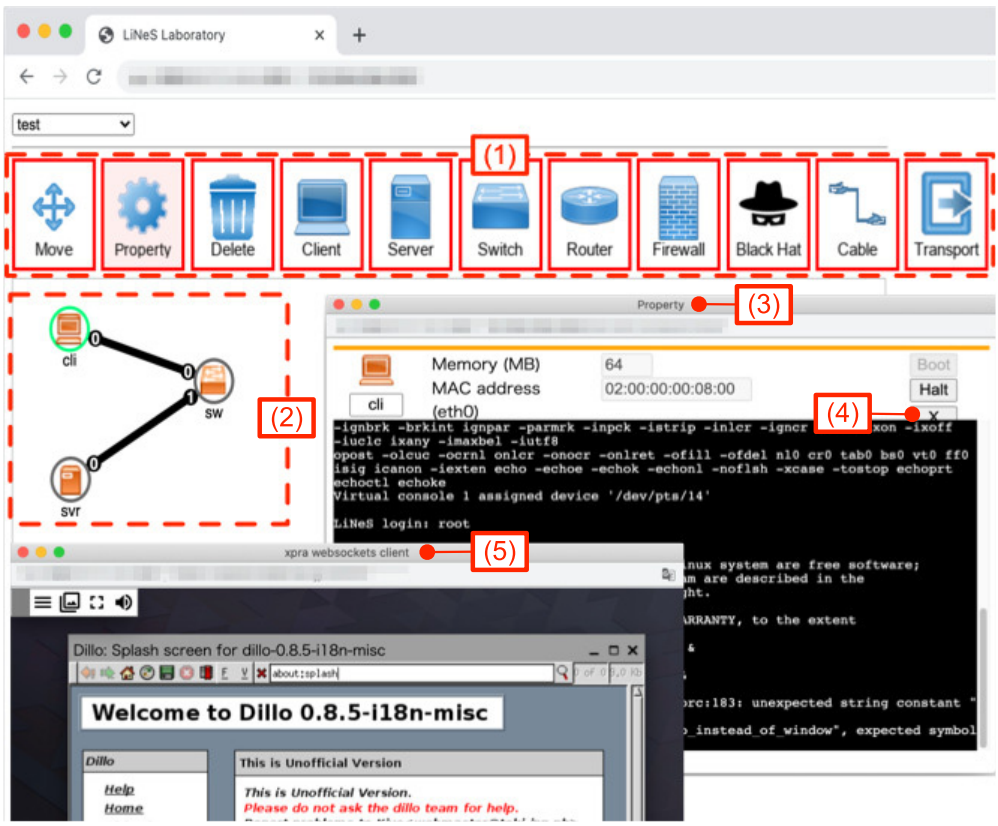


Fig. 3 Example of web pages for managing networks.

plemented with Xterm.js [18], the terminal server is implemented with socket.io [16], the remote command reception in the simulated servers is implemented with openssh [14], the Ethernet tunnels are implemented with vtun [19], and the X forwarding server and X pages are implemented with Xvfb [20] and xpra [21].

Figure 3 shows examples of clients in Google Chrome for Mac, namely a topology page, a terminal page, and an X page sorted from the top. After selecting a device from the available icons (1), clicking on the area (2) will draw the selected device. The window (3) displays the property of the device *cli* and the terminal area at the bottom draws a UML terminal running on a simulation server and accepts keyboard input. Clicking the button (4) displays the window (5) that shows the X window manager of the device *cli*. In window (5), the browser (X client) started in the device *cli* is displayed. Furthermore, the icons representing the devices in the area (2) can be moved by mouse dragging.

7. Questionnaire Survey

We carried out an experiment for evaluating usability and performance of the proposed system. There were 11 subjects in total, three of whom had already used the older version of the exercise system (Fig. 1) in the considered exercise and two of these three subjects had been teaching assistants in the exercise using the older version. In the experiment, the subjects solved exercise problems described in next sub-section with the proposed system and then answered a questionnaire.

7.1 Exercise Problems

In the exercise problems, the subjects execute the basic work in the considered exercise. Each exercise problem contains some of the elements listed below, depending on the objectives of the exercise. Some elements are concretized by values (e.g., set the IP address of server A to 192.168.0.1) or by conditions (e.g., determine the IP address of server A at network address 192.168.0.0/24). In the exercise problem shown in Fig. 4, the network topology that the subjects must build is shown at the beginning, and an example of how to set subnet masks and IP addresses is given in step 1.

- Network topologies that must be built
- Subnet masks and IP addresses that must be assigned
- Communications that must be established
- Examples of building procedure (e.g., file edit and command execution)

The subjects solve the exercise problems by the following operations.

- Placement and removal of servers, clients, switches, routers, and firewalls; connection and disconnection of cables in topology pages.
- Execution of Linux commands in terminal pages.
- Execution of wireshark [25] in X pages.

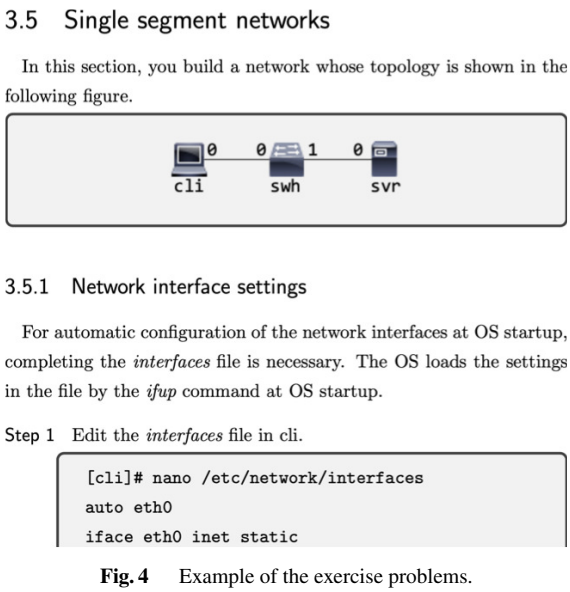


Fig. 4 Example of the exercise problems.

7.2 Questionnaire

The questionnaire consists of free comment sections for asking good points and points that should be improved on the proposed system. The subjects have to write one or more comments on each point. We extracted the comments closely related to the system features from the collected ones on each point, and then categorized them by the target of evaluation. Table 3 shows the summaries of comments in each category.

The comment 1 indicates that at least three subjects were easily able to exercise with the PCs and networks in their home. As for the reason behind this observation, they cited the accessibility using browsers. It can be said that the feature 1 of the proposed system is useful for the exercise.

The comment 2 is related to the visualization of network topologies on the topology pages (e.g., the area (2) in Fig. 3). Additionally, we received an interesting comment "The visualization will be able to help me to discover mistakes made by students." from a participating teaching assistant experiencer. This is because he/she supported students using the older version (Fig. 1) that did not display devices and connections in networks explicitly when he/she was a teaching assistant. The comment 3 means a positive evaluation for editing operations of network topologies in the topology pages (e.g., the area (1) and (2) in Fig. 3). Furthermore, we received a positive comment "I was able to build networks with fewer mistakes in shorter time by using the topology pages than by executing commands in the older version." from a subject using the older version. The functions satisfying the comment 4 are in our future plan. We will satisfy No.5 in future by implementing functions for preventing the windows from being hidden as well as possible. For example, while keeping the window size for topology pages so small as to showing network topologies (e.g., the area (2) in Fig. 3), the functions display new win-

Table 3 Questionnaire result

No.	Summary of comments	Count
1	It was nice to be able to use the system at home easily.	3
2	It was nice to be able to see cable connections and power status of the devices at a glance.	9
3	I was able to do placement/removal of devices and connection/disconnection of cables by mouse operation easily.	5
4	I hope that the system provides right-click context menu to operate networks, too.	2
5	I would like to find the windows of the terminal pages and the X pages more quickly.	3
6	I hope that the system displays IP addresses and subnet masks on the editor.	1

dows in the created blank area.

The feature 2 of the proposed system was highly evaluated in the comments 2 and 3. Furthermore, while the improvement requests (No.4 and 5) accept the feature 2, the different approaches such as the operations in the older version of the exercise system were not proposed by the subjects. Additionally, while the subjects requested the functions for finding windows more quickly (No.5), they did not request the improvements caused by mis-selection of windows. This indicates that the feature 2 solved problem (2).

While the function in the comment 6 can help students to proceed the exercise efficiently by using its output for the communication parameters, we regard displaying IP addresses and subnet masks by command execution as a part of the exercise needed for beginners. We would like to implement it together with the function that can toggle it between enabled or disabled depending on the progress of students.

The subjects exercised in their homes or the rooms of our university on their own time. Therefore, we divided the time periods in which the subjects were expected to exercise into four categories (8:00-12:00, 12:00-16:00, 16:00-20:00, and 20:00-24:00), and asked the subjects to measure network speed from their clients to the exercise server on their own timing in each category. As a result, the minimum download speed in the measurement was 3.64 Mbps, and the minimum upload speed was 8.11 Mbps. The measurement result and the questionnaire result suggest that using the proposed system at the network speeds does not leave the students frustrated.

8. Performance Measurement

8.1 Measurement Environment

The outline of the network presented in Fig. 5 is described below. The performance of the proposed system was measured in this network. The exercise server in the figure works as a hypervisor of virtual machines and employs the network shown in Fig. 6.

- Exercise server

  CPU Ryzen 9 3900X
  Memory 128 GB

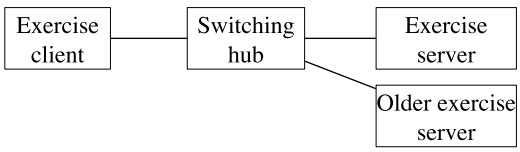


Fig. 5 Experiment network

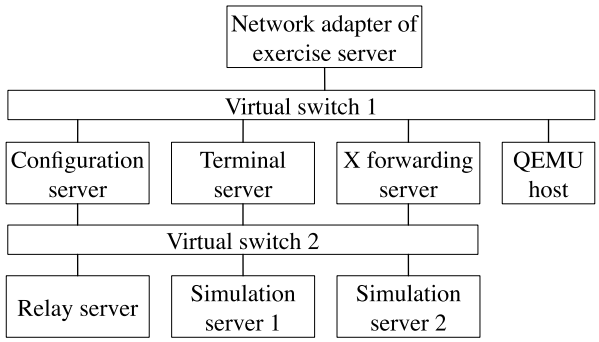


Fig. 6 Network in the exercise server.

  Hypervisor VMware ESXi-7.0

- Older exercise server

  CPU Intel Xeon E3113 3.00 GHz
  Memory 24 GB
  Hypervisor VMware ESXi-5.0

- Exercise client

  CPU Intel Core i5-7200U 2.50 GHz
  Memory 8 GB
  OS Windows 10 Pro
  Browser Google Chrome 95.0.4638.54
  vSphere Client vSphere Client 5.0.0.37933

- Network between the exercise client, the exercise server, and the older exercise server

  Specification Gigabit Ethernet

8.2 Response Time

After accessing the topology page in the exercise server (Fig. 5) by the browser running on the exercise client, the experimenter measured the response time (in seconds) spent for the operations (Table 4). The cable (local) implements connections of virtual devices running on a single simulation server, while the cable (remote) implements connections of devices running on different simulation servers. The experimenter measures the period with a stopwatch from starting the operations to noticing responses. The response times for booting the client and server start from giving the command to boot and end on displaying the head of OS startup messages on their terminal pages. The response times for halting the client and server start from giving the command to halt and end on displaying the tail of the OS stop message. The response times for placing and removing devices start from giving the commands in the area (2)

of Fig. 3 and end on confirming results. For example, after selecting a device or cable in the area (1) of Fig. 3, the measurement is started when the mouse is clicked in the area (2) and it ends when the corresponding icons are displayed.

Although the client and the server are implemented using UMLs, the time spent for placement and removal differs greatly. The reason for this is that the processing of the client involves setting up the X window system on the X forwarding server. The placement and removal of cables (remote) take more time than those of cables (local) because tunnels are built between the relay server and the simulation servers.

Table 4 Response time for operations

Object type	Placement	Removal	Boot	Halt
Client	3	5	1	9
Server	1	2	1	9
Switch	1	2	1	1
Cable (local)	1	1	-	-
Cable (remote)	7	5	-	-

Table 5 Measurement result

No.	Operation	Section	Traffic (bytes)	Time (sec.)
1	**Press enter key in terminal of proposed system**	W-T	412	0.012
		T-S	478	0.011
2	**Press enter key in terminal of older version**	W-O	13252	0.076
3	**Press enter key in terminal of QEMU**	W-T	8409	0.083
4	**Display man page in terminal of proposed system**	W-T	1383	0.178
		T-S	3366	0.177
5	**Display man page in terminal of older version**	W-O	116177	0.203
6	**Display man page in terminal of QEMU**	W-T	36885	0.598
7	**Place client**	W-C	5744	2.750
		C-S & C-X	15992	2.686
8	**Remove client**	W-C	3779	3.928
		C-S & C-X	23784	2.870
9	**Boot client**	W-C	1338	0.439
		W-T	6453	0.051
		C-S	1338	0.429
		T-S	N/A	N/A
10	**Halt client**	W-C	2222	8.184
		C-S	3672	8.182
11	**Place switch**	W-C	5645	0.435
		C-S	7702	0.380
12	**Remove switch**	W-C	3842	1.058
		C-S	7942	0.388
13	**Boot switch**	W-C	1054	0.399
		C-S	7702	0.383
14	**Halt switch**	W-C	1113	0.419
		T-S	7776	0.388
15	**Place server**	W-C	5857	0.839
		C-S	7958	0.774
16	**Remove server**	W-C	3842	0.801
		C-S	15694	0.754
17	**Connect cable (local)**	W-C	8931	1.081
		C-S	7800	0.390
18	**Disconnect cable (local)**	W-C	5792	0.506
		C-S	7734	0.400
19	**Connect cable (remote)**	W-C	9434	7.094
		C-S & C-R	86350	6.550
		R-S	10230	6.164
20	**Disconnect cable (remote)**	W-C	5792	4.291
		C-S & C-R	85772	4.209
		R-S	0	0

8.3 Traffic

We measured the communication traffic and time for the operations shown in Table 5. The measurement sections are described by "client"-"server" format. The roles are replaced by the exercise client and the older exercise server (denoted by W and O) in Fig. 5 and the initial letters in Fig. 6 (**C**onfiguration, **T**erminal, **X** forwarding, **R**elay, **S**imulation, and **Q**EMU). Tcpdump used for the measurement runs on the clients.

The measurements 1 - 6 evaluate the responsiveness of terminals in the proposed system by comparing to the other systems. In the measurements 2 and 5, the experimenter carries out the operations on the older exercise server through vSphere Client running on the exercise client. In the measurements 3 and 6, the experimenter executes the operations on QEMU 2.11.1 running on the QEMU host through VNC Viewer 6.21.118 of Real VNC running on the exercise client. The measurements 1 - 3 start when tcpdump running on the clients captures the first packet sent by pressing the enter key in the state of displaying a prompt on the terminals like Fig. 7, and they end when the tcpdump captures the packet storing the tail letter of next prompt. The measurements 4 - 6 start when tcpdump captures the first packet sent by pressing the enter key after typing 'man man' on the terminals, and they end when the tcpdump captures the final packet used for displaying manual pages of man. The result indicates that the amounts of the packets captured in the proposed system are fewer and the times while the packets were captured in the proposed system are shorter than those of the other systems. We consider that the reason is that the contents of the terminal in the proposed system are text format, while the one in the other systems are image format.

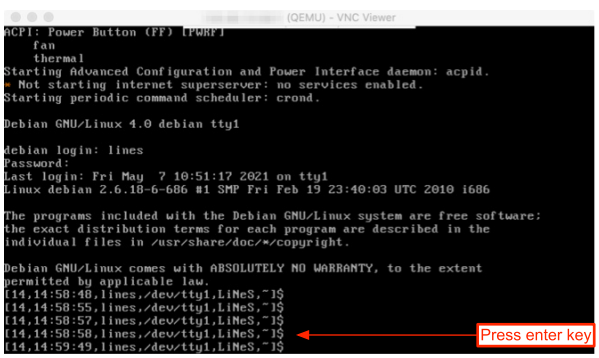


Fig. 7 Terminal in QEmu

Therefore, we conclude that the narrower the bandwidth of the network is, more lightly the proposed system works than the other systems do.

The measurements 7 - 20 evaluate system performance of the proposed system. The section 'T-S' of No.9 has no values because the experimenter was failure to estimate the packets storing the head of OS startup messages due to encryption of the payloads. Because the boot and halt of servers work in a similar way to those of the clients, we describe only those of clients in the table (No.9 and 10). There is no traffic (the section 'R-S' in No.20) between the relay server and the simulation servers because the tunnel client processes are terminated by the 'SIGKILL' signals without negotiations to the tunnel servers. The traffic generated by each operation in the section "W-C" is nearly lighter than one generated by pressing enter key in the other systems. This fact introduces that the proposed system is able to provide the exercise environment where students configure networks by command line, not X window applications, even in the networks as poor as the other systems are not able to do it adequately.

9. Quality of Experience Assessment

9.1 Assessment Method

We carried out the experiment in which sixteen subjects evaluated the responsiveness of the system by the double-stimulus impairment scale (DSIS) method [22]. In the experiment, each of the subjects accessed to the server from the client shown in Fig. 8, and the subject evaluated responsiveness for each of the operations shown in Table 6 as follows:

1. The subject performed an operation without bandwidth limitation.
2. The subject performed the same operation with bandwidth limitation.
3. The subject assessed the degree of deterioration of the responsiveness in step 2 based on that in step 1 using the following five-grade impairment scale: 5 - imper-

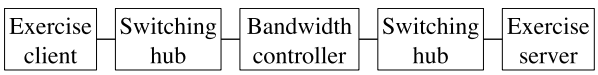


Fig. 8 Network for QoE assessment.

Table 6 Operations in QoE assessment

ID	Summary
A	Operations for network topology 1. A subject creates the network topology similar to one shown in the area (2) of Fig. 3. 2. The subject opens the property of each device and then boots them.
B	Operations in terminal 1. A subject opens the specified file in the terminal page of the server with the nano editor [23]. 2. The subject changes the specified words written in the second page of the file.
C	Operations in X window system 1. A subject starts xterm [24] in an X page. 2. The subject performs steps 1 and 2 of operation B in the terminal.
D	Operations for wireshark [25] in X window system 1. A subject starts wireshark in an X page. 2. The subject starts to capture packets with the wireshark. The captured packets are listed on the area (1) in Fig. 9. 3. The subject stops capturing packets when capturing more than 2000 packets. 4. The subject selects the packet of No. 100 and confirms its size displayed in the area (2). 5. The subject selects the packet of No. 1000 and confirms its size.

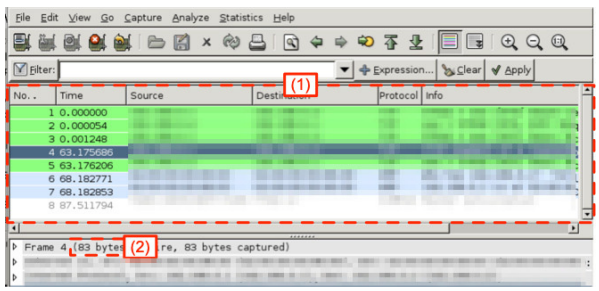
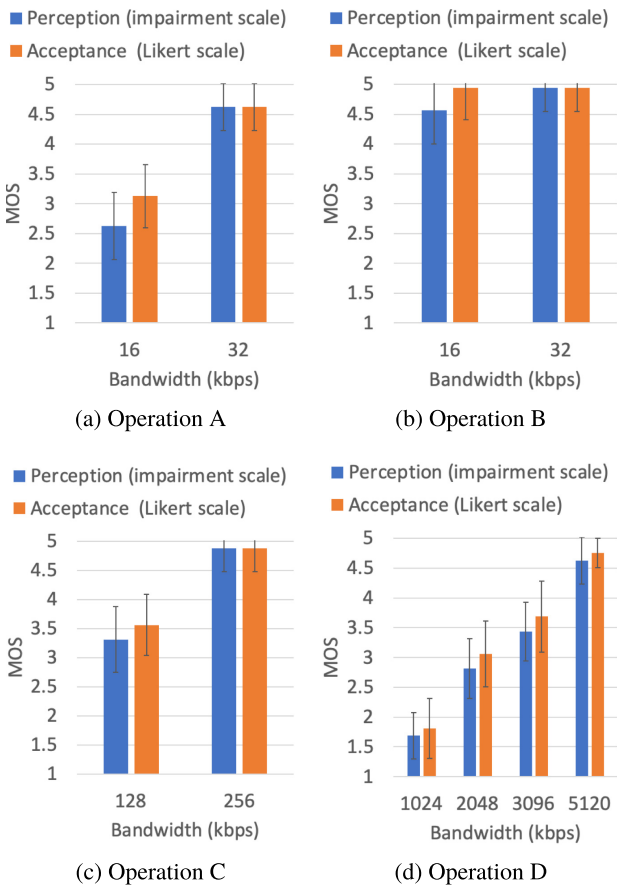


Fig. 9 Wireshark for operation D.



(a) Operation A (b) Operation B

(c) Operation C (d) Operation D

Fig. 10 MOS of responsiveness evaluation for topology operations.

ceptible, 4 - perceptible but not annoying, 3 - slightly annoying, 2 - annoying, 1 - very annoying.
4. The subject answered the degree of acceptance of the responsiveness for the operation in step 2 using the following five-point Likert scale: 5 - very much, 4 - a lot, 3 - somewhat, 2 - not much, 1 - not at all.

The operations in Table 6 are typical in the exercise and cause various responses from the system: X windows are refreshed on scrolling and clicking by mouse operations and on viewing execution of monitoring tools; terminals are refreshed on moving a cursor, inputting letters, and transiting pages by keyboard input.

9.2 Experimental Results

In Fig. 10, the x-axis stands for upstream and downstream bandwidth of the network, and the y-axis stands for the mean opinion score (MOS), and the error bars show a 95% confidence interval. We see from Figs. 10(a) and 10(b) that the exercises in which students configure networks by command line, not X window applications, requires at least 16 kbps bandwidth and more than 32 kbps for comfortable. From Fig. 10(c), it can be said that the operations which cause refreshing views with low rate require at least 128 kbps bandwidth and more than 256 kbps for comfortable. On the other hand, Fig. 10(d) indicates that the operations causing high-rate refreshing require at least 2048 kbps bandwidth and more than 5120 kbps for comfortable.

10. Conclusion

We proposed a novel network security exercise environment for students meant to efficiently use UML networks running on remote servers. It implements virtual networks by connecting UMLs running on a single server with Linux Bridges, and connecting UMLs running on different servers with Linux Bridges and Ethernet tunnels. The students manage the networks via web pages for editing the connections among virtual devices, for input and output with UML terminals, and for input and output with UML X clients.

We carried out three evaluation experiments. In the first one, we asked the subjects for comments on the usability of the proposed system after carrying out a simple exercise. In the second experiment, we measured the response time and traffic for the main operations as system performance. Because the traffic is mostly small, it is expected that users not really notice the deterioration in responsiveness even while accessing to the servers through narrow band networks. In the third experiment, we evaluated quality of experience (QoE) of the responsiveness of the system on various network bandwidth. The result of the experiment showed that the exercises using nano in the terminal pages required at least 16 kbps bandwidth and ones using wireshark in the X pages required at least 2048 kbps bandwidth.

Future work includes the improvements of the operability pointed out by the subjects. In addition, we have a plan to implement functions for collecting operation history of users in order to do learning analytics. Especially, high accuracy analytics are expected because the function will be able to collect input operations on terminals character by character.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 20K12108. The authors thank Professor Yutaka Ishibashi for technical advice on QoE assessment.

References

[1] J. Dike, User Mode Linux, Pearson, 2006.
[2] VMware, vSphere Single Host Managemen, https://docs.vmware.com/en/VMware-vSphere/5.5/com.vmware.vsphere.hostclient.doc/GUID-52A4C8B5-04F9-4571-9AC3-4FBED2DD9215.html, 2019.
[3] Y. Tateiwa, "Development of web-based hands-on system for network security classes with intuitive and seamless operability and light-weight responsiveness," IPSJ SIG Technical Report, vol.2021-CLE-33, no.11, pp.1–6, March 2021 (in Japanese).
[4] Y. Tateiwa and N. Iguchi, "Hands-on laboratories for network administration exercises based on virtual machine technology," The Journal of the Institute of Electronics, Information and Communication Engineers, vol.104, no.8, pp.872–878, Aug. 2021 (in Japanese).
[5] Y. Tateiwa, "Evaluation of web-based hands-on system for network security classes with intuitive and seamless operability and light-weight responsiveness," IEICE Technical Report, vol.121, no.294, pp.1–6, Dec. 2021 (in Japanese).
[6] N. Iguchi, "Development of a self-study and testing function for NetPowerLab, an IP networking practice system," Int. J. Space-Based and Situated Computing, vol.4, no.3/4, pp.175–183, 2014.
[7] L. Xu, D. Huang, and W.-T. Tsa, "Cloud-based virtual laboratory for network security education," IEEE Trans. Education, vol.57, no.3, pp.145–150, Aug. 2014.
[8] CloudLab, https://www.cloudlab.us/, accessed Nov. 1, 2021.
[9] ThoTh Lab, https://www.thothlab.com/, accessed Nov. 1, 2021.
[10] R. Beuran, D. Tang, C. Pham, K. Chinen, Y. Tan, and Y. Shinoda, "Integrated Framework for Hands-on Cybersecurity Training: CyTrONE, Computers & Security," vol.78, pp.24–35, 2018.
[11] KVM, http://www.linux-kvm.org/page/Main_Page, accessed March 23, 2022.
[12] Cloud Computing Services - Amazon Web Services (AWS), https://aws.amazon.com/, accessed March 23, 2022.
[13] Node.js, https://nodejs.org/en/, accessed Nov. 1, 2021.
[14] OpenSSH, https://www.openssh.com, accessed Nov. 1, 2021.
[15] SVG.js v3.0 - Home, https://svgjs.dev/docs/3.0/, accessed Nov. 1, 2021.
[16] Socket.IO, https://socket.io/, accessed Nov. 1, 2021.
[17] jQuery, https://jquery.com/, accessed Nov. 1, 2021.
[18] Xterm.js, https://xtermjs.org/, accessed Nov. 1, 2021.
[19] VTunVirtual Tonnels over TCP/IP networks, http://vtun.sourceforge.net, accessed Nov. 1, 2021.
[20] XVFB, https://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml, accessed Nov. 1, 2021.
[21] xpra home page, https://xpra.org, accessed Nov. 1, 2021.
[22] ITU-R Recommendation BT.500-13, "Methodology for the subjective assessment of the quality of television pictures," Jan. 2012.
[23] The GNU nano homepage, https://nano-editor.org/, accessed March 18, 2022.
[24] XTERM - Terminal emulator for the X Window System, https://invisible-island.net/xterm/, accessed March 18, 2022.
[25] Wireshark Go Deep., https://www.wireshark.org/, accessed March 18, 2022.

Yuichiro Tateiwa received his Ph.D. in Information Science from Nagoya University in 2008. He is interested in utilizing virtual machine technology for education.