A Thesis

entitled


by

Wei Liu

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in

Engineering with Concentration on Computer Science

Dr. Weiqing Sun, Committee Chair

Dr. Ahmad Javaid, Committee Co-Chair

Dr. Hong Wang, Committee Member

Dr. Amanda Bryant-Friedrich, Dean
College of Graduate Studies

The University of Toledo

August 2016
An Abstract of


by

Wei Liu

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Engineering with Concentration on Computer Science

The University of Toledo

August 2016

In this thesis, we implement and deploy a light-weight, flexible, and web-based testbed to host multiple virtual networks on a single underlying physical network infrastructure for network and security experiments in an academic set-up. Creating and deleting virtual machines (VMs) and virtual networks (VNs) are time-consuming tasks. To make those tasks easier, much virtualization software is used by universities and institutions that enable the VMs and VNs creation and deletion efficiency on a single physical infrastructure. However, these technologies are heavy-weight. In contrast, we use Linux Containers (LXC), Open vSwitch (OVS), and Shell In A Box (ShellInABox) to deploy a light-weight and user-friendly web-based system. A user-friendly browser-based system is implemented that replaces the traditional terminal-based approach to access VNs, which makes the testbed easier to use. The browser-based system provides interactive access to the virtual hosts via a gateway machine within a web page. By login to the web-based system administrator can create or delete virtual networks automatically based on the available resources on the participant physical hosts determined by the algorithm running on the backend of the testbed.
# Table of Contents

Abstract ........................................................................................................................................ iii

Table of Contents ......................................................................................................................... iv

List of Figures ............................................................................................................................... vi

List of Abbreviations ..................................................................................................................... viii

1 Introduction................................................................................................................................... 1

1.1 Introduction .............................................................................................................................. 1

1.2 Related Work ............................................................................................................................ 4

2 Design and Implementation of the testbed ................................................................................... 8

2.1 Design and Implementation of the front-end of the testbed .................................................... 9

2.1.1 User Interface for Student and Administrator ................................................................. 9

2.1.2 User design framework ...................................................................................................... 11

2.2 Implementation of the back-end of the testbed ..................................................................... 14

2.2.1 Linux Containers (LXC) .................................................................................................. 15

2.2.2 Open vSwitch (OVS) ...................................................................................................... 16

2.2.3 Back-end Implementation ............................................................................................... 16

3 Web-based UI for testbed access ............................................................................................... 22

3.1 Student User Interface ............................................................................................................ 22

3.2 Administrator User Interface ................................................................................................ 28
3.2.1 Add a virtual network ..............................................................29
3.2.2 View physical host ..................................................................31
3.2.3 View/Delete a virtual network ..................................................32

4 System performance and evaluation ..............................................35
  4.1 System performance ..................................................................35
  4.2 Discussion ..................................................................................38

5 Conclusion and Future work ..........................................................41

References ......................................................................................43
List of Figures

1-1 Virtual Machine Monitor ........................................................................................................3
2-1 Light-weight Testbed Front-end and Back-end ................................................................. 8
2-2 Front-end Design ................................................................................................................... 9
2-3 ShellInABox Result .............................................................................................................. 10
2-4 Administrator Subsystem Design ..................................................................................... 11
2-5 Struts Workflow .................................................................................................................. 11
2-6 Spring Framework .............................................................................................................. 12
2-7 Hibernate Framework ........................................................................................................ 13
2-8 Administrator Subsystem in ESXi ..................................................................................... 14
2-9 LinuX Containers and OpenvSwitch ................................................................................. 15
2-10 LinuX Containers Infrastructure ..................................................................................... 16
2-11 Virtual Host and Virtual Switch Created by VMWare ESXi ........................................... 17
2-12 Create New Virtual Machine by Clone Standard Virtual Machine ............................... 18
2-13 Virtual Machine Connect OVS Communication .............................................................. 18
2-14 GRE Tunneling between two Hosts Tunneled with the remote IP ............................... 19
2-15 Testbed with two Virtual Networks ................................................................................. 19
3-1 UML Sequence of User System ....................................................................................... 22
3-2 Testbed Student Subsystem Login Flowchart ................................................................. 23
3-3 Testbed Student Subsystem Login User Interface ........................................24
3-4 Testbed Student Subsystem Main User Interface ......................................24
3-5 Testbed Student Subsystem Instruction Interface ......................................25
3-6 Testbed Student Subsystem Gemini Interface ..........................................26
3-7 Testbed Student Subsystem Virtual Host Access UML Sequence .................27
3-8 Testbed Admin Subsystem Add Virtual Network Front-end .........................29
3-9 Topology Commands ..................................................................................30
3-10 Topology Created by Command ..................................................................30
3-11 DefaultGW and DMZ Virtual Host Information in Testbed .......................31
3-12 Testbed Admin Subsystem Virtual/physical Host Information ....................32
3-13 Virtual/Physical Host Information in Testbed ...........................................33
3-14 Virtual/Physical Host Information Flow ....................................................33
4-1 Course Assignment Network Topology .......................................................34
4-2 Brand Width Per Host (Mbps) .....................................................................35
4-3 CPU Usage ..................................................................................................36
4-4 Memory Usage ............................................................................................36
4-5 Latency Relation to Number of Virtual Hosts .............................................37
4-6 Experience Rate in Students .......................................................................38
List of Abbreviations

GRE.........................Generic Routing Encapsulation
IP.............................Internet Protocol
ICMP..........................Internet Control Message Protocol
LightVN......................Light Virtual Network
LXC............................Linux Containers
MLN............................Multiple Logical Node
NFS..............................Network File System
OVS..............................Open vSwitch
OS...............................Operating System
SSH..............................Secure Shell
ShellInABox.................Shell In A Box
VLAN..........................Virtual Local Area Network
VMs.............................Virtual Machines
VMM.............................Virtual Machine Monitor
VNs.............................Virtual Networks
Chapter 1

Introduction

1.1 Introduction

Nowadays, computers and networks security are important to personal computer users, organizations, and institution. Computers and networks are the priority for a broad variety of applications. Therefore, security is crucial to them [1]. It is necessary to study and research various issues related to computers and networks security. To better understand the security, instructors and researchers need to design and perform a series of experiments. Typically, there are two ways to establish experimental environments either using real physical systems or simulation-based systems.

Actual network traffic and real networks environment can be set up in real physical systems. Experimentation on these physical systems helps to understand and require the concepts of network security in support of new ideas. However, setting-up of a real physical based environment needs a lot of efforts and is time-consuming in an academic environment. In contrast, the simulation-based environment provided a cost-effective and high-availability test environment by using some simulators but failed to model the real world scenario problems.
There are many virtualization technologies that make the task of setting-up virtual machine easier compared to the establishment of the real-physical based environment. The virtual machine concept is becoming more and more popular in all computing areas and gradually replacing the traditional way of physical servers. The physical servers are expensive [2] and it is observed that quite often resources are under-utilized in them. Furthermore, with the increased tasks and shared resources, considerations, such as conflicts of resource and cost factors need to be taken into account [3]. In contrast to the physical servers, the advantages of virtual machines are obvious in lower equipment, power, portable, management, and hardware costs [2]. Virtual machines have independent disk storage and operating system [4]. If a virtual machine attacked by an adversary or fails in operation, it will not affect the host machine. The virtualization technology protects one virtual machine user group from the malicious activities of other unauthorized users. Besides security can also be delivered as an efficient service independent from physical hosts, integrated and constructed precisely where it is needed without any hardware updates.

All those advantages mentioned above make the virtualization technology convenient for creating and testing a network environment. It is easy to clone the workstations and safe to test. This approach saves a lot of time, effort, and cost. A bunch of virtual servers and workstations can be created or copied, and isolated in their private networks so that nothing harms the existing networks. So far, there are several virtualization tools for different operating systems, such as VMware, Virtual Box, Parallels, QEMU, Windows Virtual PC, etc. [5]. They are either available as commercial
or under free open source public license. All these conveniences of virtualization motivate many academicians from different institutes to develop testbed using virtualization.

Virtual machine monitor (VMM) is the main software used in modern virtualization technologies to manage VMs. The VMM, also called a hypervisor, provides specific system resources to each virtual machine over the underlying host Figure (1-1).

The hypervisor intercepts the guest calls from the VMs and translates them into the appropriate system calls to the host system. All VM's can get access to CPU, memory, persistent storage, I/O devices, and the network in the control of VMM [6].

However, when a VM needs a VMM to access the host resources indirectly, it brings a full-virtualization environment with kernel isolation [6]. The disadvantage of full virtualization is the heavy utilization of underlying resources and therefore poses a challenge to the scalability of the virtual environment.

To make a more scalable testbed, we propose a testbed system for network security researches and experiments, designed and deployed to host multiple virtual networks on a single physical network infrastructure using light-weight Linux Containers (LXC) to manage virtual hosts. LXC is more efficient in resource utilization for abstracting the operating system kernel and shared operating system compared to VMM that emulates virtual hardware and abstracts an entire device [7].
Browser/Server architecture is an improvement over Client/Server architecture with the rise of web technology. In this architecture, the user interface is implemented in a browser with the main business logic on the server side (Server) achieves a so-called three-tier structure. In the Browser/Server architecture, the user can fetch text, data, images, animations, video, and voice information which are generated by a web server in its local machine using a browser. The web server can connect to the database to fetch a large amount of data stored in it. Besides, clients only need a browser and can simply download the data from the web server to the local machine.

In our thesis, we provide a web-interface based front-end to access to the testbed, set the virtual networks, and use ShellInABox [5] which enables interactive terminal based access to the remote machines via a web application. ShellInABox is applied to provide remote access to the virtual hosts to execute Linux commands and scripts [8]. The proposed system consists of two user groups: 1. The student user logs in to the web-based system to perform experiments and assignments. 2. Instructors or administrators use the system for setting-up virtual networks and user groups, view the testbed status, etc.

1.2. Related Work

There have been several types of researches conducted that address the development of the testbed for an academic environment. At Iowa State University, a project named Xen Worlds was designed to set an information assurance virtualized lab environment. There are a large number of students getting access to the virtualized lab environment remotely that requires high security. Both on-campus and off-campus students use the testbed at the same time. The equally accessible and easy-to-use are the priorities for
designing this system. This virtual environment system was 24/7 online system equally accessible to on- and off-campus students. According to curriculum requirements, Xen Worlds can set-up the various experimental environments for students [9] [10].

Manage Large Networks (MLN) can implement a predefined or student-designed virtual network of arbitrary complexity. This MLN environment is a low-cost, scalable load-balancing Linux cluster. The instructor can use MLN environment to set an effective networking lab [11] [12].

Virtual network laboratory architecture can support laboratory exercises and student’s network assignments. Instruction of learning goals with the course syllabus and related knowledge and additional skills evaluation criteria for the laboratory exercises can be set in a laboratory or assignment environment for students. Lessons learned and practiced by using the virtual network laboratory can help students to master the required knowledge easily and acquire hands-on experience. The instructor can analyze the students’ skills and estimate strong and weak skills after evaluating their assignment through the system [13] [14].

Mininet can prototype large networks on the constrained resources of a single laptop rapidly. The Mininet virtualization is a lightweight approach of using OS-level, including processes and network namespaces, allows it to copy or clone to hundreds of child nodes. Mininet supported case studies and developed Software-defined Network [15] by over 100 users, at 18 institutions. Mininet enables self-contained prototypes to be downloaded, run, evaluated, explored, tweaked, and built the personal computer or laptop [16].
V-NetLab can separate virtual networks of computers. It can simplify virtual network start-up and shutdown processing. It provides a virtual network interfaces for configuring different physical or virtual hosts. These virtual networks are realized on modest physical resources [17] [18].

A cloud-based system named V-Lab is a contained experimental environment for students or researcher’s experiments. This system uses visualization technologies and virtual switches. Students or researchers can perform tasks remotely. V-Lab provides a detailed learning path for network security hands-on skills. This system helps students or researchers to understand and build network security knowledge [19] [20].

The University of Toledo provides LightVN, a virtual environment, for teaching and research on computer networks and security. Virtual networks can be set up on a single physical infrastructure by LightVN. This OS-level virtualization makes our system cost-effective, scalable, and maintainable [21].

Naval Academy provides a simulation environment for attacks on computer systems [22]. This system has Internet attack simulator and a network simulator. It can launch various Internet attacks against the virtual networks. Students can acquire cyber security and network knowledge [23].

User-mode Linux (UML) provides virtual system environment for server integration and testing buggy software. UML has functions for comprising networks. It trains and teaches students to get knowledge of network security on virtual networks realized through single physical computer [24].

Virtual networking laboratory (VNLab) provides interfaces to get access to virtual networks and real networking devices. In Cisco Networking Academic experiment, this
system usually has been used. Students and researchers can perform tasks and experiments in this system [25].

NICE [26] means Distributed vulnerability detection, measurement, and countermeasure called It can monitor and control virtual programmable switches to improve the virtual system network security. Using network programming API can enhance the existing security network. Therefore, the system designer can build experiment environment for students and instructors more flexible.

The systems mentioned above have limitations regarding scalability and user interfaces. LightVN is a light virtual network testbed that uses less physical machine resources, but it lacks in flexible virtual network set-up and automation. Other systems are heavy-weight virtual network system. They need more resources to realize the system. In contrast, our testbed provides automatic startup and shutdown of these administration virtual networks with a simplified processing. This approach can modify and set networks configuration in a modest size physical networks consisting of commodity personal computers.

The structure of thesis is as follows. Chapter 2 describes the design and implementation of the testbed. Chapter 3 discusses Web-based UI for testbed access for students and administrator/instructors. Chapter 4 evaluates the performance of the testbed and provides the student feedback. Finally, Chapter 5 concludes our research and outlines the future work.
Chapter 2

Design and Implementation of the testbed

In this chapter, we discuss the design and implementation of our light-weight testbed. From the architecture, the testbed has two parts: Front-end that deals with the user interface to access the testbed and the back-end deals with internal implementation and set-up of the testbed. Figure (2-1) shows the overall architecture of the proposed testbed.

![Diagram of testbed architecture]

Figure (2-1): Light-weight Testbed Front-end and Back-end
We briefly discuss the front-end with an emphasis on the implementation of the back-end of the testbed. We discuss the user-interface in detailed in Chapter 3.

2.1 Design and Implementation of the front-end of the testbed

2.1.1 User Interface for Students and Administrator

The user interface system is implemented using Java web technology including Struts [27], Spring [28], and Hibernate [29] as the front end shown in Figure 2-2. The user can use a browser to communicate with the Java web container Apache Tomcat through HTTP protocol. Using Browser/Server architecture, users can check their experiment or the status of the network either through a PC or a smartphone browser. Comparing to the Client/Server PC desktop software such as Putty [30], users can get access to their networks by any browser.

![Diagram of front-end design](image)

Figure (2-2): Front-end Design

In order to support Browser/Server architecture, we need a Linux command server to get a request from user’s browser and return the virtual machine’s response to it. We use ShellInABox, an open source, reliable, and well-documented web-based Linux command server and tweak it as per our requirement. ShellInABox sits in the middle
layer. It sends Linux command to virtual machines and returns the result to the users through the HTTP tunnel.

We install ShellInABox and Apache Tomcat in one machine which we refer as Gateway since it separates the public network from our private testbed network. Apache Tomcat can send Linux command to ShellInABox. And, then ShellInABox will forward the command to virtual machines. The result will be sent back to Tomcat. In this way, users can get virtual machine response Figure (2-3).

![Figure (2-3): ShellInABox Result](image)

The administrator part of the user interface is implemented by JSP/Servlet. It can manage each user group virtual network through the user-friendly interface management environment. We implement a Java server program as that uses Linux commands at the back-end on each participating hosts. A Java client program runs on the Gateway that sends commands to all the servers for managing the testbed from the web system.
The administrator can add new virtual networks, delete virtual networks, and view the existing participating host’s status Figure (2-4).

**2.1.2 User design framework**

Model–view–controller (MVC) is one of the features of the J2EE. Struts inherits MVC to develop Java web applications. We design our system front-end using Struts, Hibernate, and Spring.
Users send the request to struts framework web container. Struts main controller will intercept all users’ request. After loading configuration file, the main controller forwards the request to Action. The users’ request information will be encapsulated in From Action. After handling the From Action, Action will return the result to the main controller. A View file will be presented decided by main the controller configuration file Figure (2-5).

In Figure (2-6), the Spring Framework has multi-layer J2EE architecture frameworks base on Inversion of Control (IoC) or Dependency Injection (DI) and Aspect Oriented Programming (AOP). It is a very popular open source framework. Spring IoC can create an object and inject an object into another object in a passive way compared to creating or finding the dependent objects. All objects are java beans created and destroyed through Spring container. Spring container will control the object life cycle. IoC based on reflection programming and factory method. Using XML file, spring can get class
information of the class definition. According to the corresponding object class name given in XML, the reflection programming generates the object. Aspect Oriented Programming (AOP) can be used in many program processing. In our work, we mainly use Spring AOP to manage transaction processing and logging.

In Figure (2-7), Hibernate open source object-relational mapping (ORM) framework is the middle layer or persistence layer between Java Object Model and relational database management system. Using Hibernate, the object model can store data into the database without writing SQL. Hibernate API encapsulates JDBC, Java Transaction, and Java Naming and Directory Interface. Hibernate can be integrated with Tomcat using Java Naming and Directory Interface (JNDI) and Java Transaction API (JTA). In our system, we connect and manipulate MySQL through Hibernate.

![](image)

Figure (2-7): Hibernate Framework [29]

### 2.2 Implementation of the back-end of the testbed

The back-end architecture is the challenging part of our testbed. The back-end system provides separate virtual networks to different student user groups. To do this, the back-end system must meet several requirements. To accommodate multiple user groups simultaneously, the virtual hosts should be light-weight. Linux Container (LXC) is used to create or delete a new virtual host for each user group that makes the testbed
light-weight compared to others. Each group’s virtual network can co-exist with the others in the same infrastructure without interfering each other. Also, each virtual network’s traffic must be isolated from others. Open vSwitch (OVS) is used to isolate each virtual network while hosting them on the same infrastructure. Virtual network configurations of one group should not affect the other groups such as firewall, etc.

VMWare ESXi, virtualization software, can set-up VMs and virtual switches in a single physical machine. We use it to create VMs to replace the need of physical hosts. We use VMWare ESXi to create virtual machines and virtual switches shown in figure 2-8.

![Figure (2-8): Administrator Subsystem in ESXi](image)

We use LinuX Containers (LXC) as virtual machine container. We can create several virtual hosts using LXC. From now onwards, we will use a virtual machine for physical hosts and virtual hosts for LXC based virtual machines to avoid the confusion in
further sections. We use VLAN feature of Open vSwitch (OVS) to separate virtual networks traffic. Figure 2-9 shows the LXC and OVS together.

![Diagram of LXC and OVS](image)

Figure (2-9): LinuX Containers and OpenvSwitch

### 2.2.1 LinuX Containers (LXC)

LinuX Containers (LXC) is a Linux container interface for user’s virtual host. In LXC, cgroups can manage CPU, memory usage, disk I/O, and separate file system, network isolation, and privilege separation to share and allocate host resources. LXC can simulate a complete Linux system without any extra hardware. LXC also can separate the virtual hosts totally from the host machine. LXC provides isolation and security environment. The host machine and virtual hosts on it separate with each other. Even a hacker get access to the virtual hosts, it cannot affect the host machine without authorization. Compared to other virtualization tools, LXC use very less host machine resources, but highly-efficient. Users can focus on their experiment rather than tedious, time-consuming virtual networking settings and separation of the networking traffic. All these features make LXC a good candidate tool for our testbed. In Figure (2-10), there are
four Linux virtual hosts installed using LXC. The container manager can add or delete LXC, can allocate resources, such CPU, memory, and disk space to virtual hosts.

![Diagram of Linux Containers Infrastructure]

Figure (2-10): Linux Containers Infrastructure

### 2.2.2 Open vSwitch (OVS)

Open vSwitch (OVS) is an open source software that provides network automation and virtual switch at its basic functionalities. OVS can run on a different version of Linux platform and can embed into host kernel. It can be used in another Linux compatible operating system platform. OpenStack, OpenNebula, and OpenQRM have installed OVS as the software setting switches and hardware programmable switches. We integrate the VLAN with trunking and Generic Routing Encapsulation (GRE) features of OVS in our testbed.

### 2.2.3 Back-end Implementation

We create a VMWare ESXi host on a physical machine. In VMWare ESXi, we create several Linux virtual machines shown in Figure (2-11). On the same ESXi host, we
create the same number of virtual switches. The virtual machines are connected with each other using virtual switches and form a complete graph.

![Diagram of virtual hosts and virtual switches created by VMWare ESXi]

Figure (2-11) Virtual Host and Virtual Switch Created by VMWare ESXi

To create and manage virtual hosts in each virtual machine, we install LXC in each of them. We create a template virtual host LXC and use it to clone a new virtual host shown in Figure (2-12).

![Diagram of creating a new virtual machine by cloning a standard virtual machine]

Figure (2-12) Create New Virtual Machine by Clone Standard Virtual Machine
We create an OVS Bridge for each of the network interface of the virtual machine and map it accordingly. We connect the network interface of a virtual host with the OVS Bridge and assign it a unique VLAN id based on the group owner of that virtual host Figure (2-13).

![Diagram of Virtual Machine Connect OVS Communication](image)

Figure (2-13) Virtual Machine Connect OVS Communication

The virtual networks traffic is confined by GRE tunneling. In this way, the traffic of a virtual network can be isolated from other. Using GRE setting the same network configurations and GRE port on OVS Bridge, our testbed can be isolated from the public network Figure (2-14).
In Figure (2-15), we use virtual switch created by VMWare ESXi to enable the communication of one virtual host with another virtual host residing on
different machines. Gateway system is connected with other virtual machines through a management switch and can perform all the task in the testbed through a web-based system.
Chapter 3

Web-based UI for testbed access

The front-end interface is developed using HTML, JavaScript, CSS, AJAX, jQuery, JSP, Spring, Hibernate, Struts. Its session management is built upon Apache Java web application and data storage based on MySQL database. The students or administrators can use front-end web-based login interface without typing commands in terminals to remote login the virtual networks and SSH the virtual hosts. User groups can get access to their accounts through web pages and SSH into different virtual hosts using web pages, and then send Linux commands on a web based terminal interface to run them on the testbed, and get the result from virtual hosts. The benefit of this web-based access is that it eliminates the tedious operations of typing command in different terminals. The system provides a web-based experience and from the perspective of users and the administrators, it comes up with a more convenient, understandable and easier way compared to the old virtual network testbed.

The most challenging part is to send a request from user’s web-based clients and get responses from virtual machines without using repetitive user authentication. Client-side terminal emulator and server-side terminal emulator are two main types of web-based terminal servers. In order to adapt various browsers, some emulators need to install plugins to complete the web-based terminal function. For instance, an extension is
required to install in the browsers. In our testbed, we choose ShellInABox as the server-side terminal emulator. It is an open source web-based server-side terminal emulator. Any JavaScript and CSS supporting web browsers can get access to ShellInABox and do not require any additional plugins.

Users follow the sequence shown in the Figure 3-1 to do their assignments on their virtual networks. Using the group identity, users can login to the web application system. After getting access to the system, users can send Linux command to Tomcat which will forward the command to Linux command server Shellinbox. Virtual hosts get a command from Shellinbox to configure themselves. The result will return to ShellInAbox, and then will return to the Tomcat. The Tomcat will send the result to the browser.

![Figure (3-1): UML Sequence of User System](image)

### 3.1 Student User Interface

We use Java web technology as mentioned earlier to develop the front-end interface. The student user interface includes four parts: login, lab instruction, terminals, and log out.
Log-in is the group identification web interface for the students. The group identity is used for authentication to prevent illegal users to login into the system. Besides, the group identity is used to get access to the Gateway, virtual machines, and virtual hosts. Using the group identity, students or admin can get access to the front end web application installed in the Gateway. In this way, students can focus on their assignments or experiments without tedious log-in processing shown in Figure (3-2).

![Flowchart](image)

Figure (3-2): Testbed Student Subsystem Login Flowchart

Students can get access to testbed by using group identity. Figure (3-3) shows the Login User interface. The user group will log in to the web application in Apache Tomcat by authenticating user group information in the MySQL database.
Figure (3-3) Testbed Student Subsystem Login User Interface

Figure (3-4) Testbed Student Subsystem Main User Interface

Figure (3-4) shows the display for a user group after correct login. In this main user interface, there are three tabs include lab instruction, terminals for each virtual host to
perform the lab assignments, and log-out. User groups can log out from the system by clicking logout icon. In this way, the web application will expire user group web session.

The virtual hosts continue to run the Linux commands given from the user group before log-out. Therefore, when the same user group logs-in again, the ShellInABox can display the last command results. User groups can remain and maintain their last time status without repeating the whole processing of their assignments or experiment. And some experiment requires the virtual hosts need to observe network traffic for a period. The testbed can fulfill this kind of task without tedious processing. Students can put more effort in their experiments rather than repeating the same processing every time.

Figure (3-5) Testbed Student Subsystem Instruction Interface

In Figure (3-5), student group can check the instruction of the lab or assignment. The instruction file can be changed according to the requirement of the experiment by uploading a new instruction file or change an existing instruction file.

The virtual hosts name such as Gemini, NFS, IntFW, ExtFW, DefaultGW, and DMZ and icons for their terminals access for a particular user group. When a user group clicks
the terminal icon, it will connect to virtual host assigned to the user group through ShellInABox. In common, there is three-time verification to get access to the user group virtual host. First, the user needs to get access to Gateway using Putty or SSH plug-in software in the browser. Second, the user has to SSH choosing virtual machine IP address with the group identity. At last, the user can get access to the virtual host assign to this user group. After completing all those steps, the user can use Linux Command to do their experiments or assignments. Besides all those verifications, users need to install putty in PC or SSH plug-in software to the browser. In contrast, users only need to click an icon in the web application and then users can get access to the virtual host assigned to that group. Users only need a browser in a computer.

In Figure (3-6), when the user clicks Gemini virtual host icon, it shows the terminal for that host in the ShellInABox. When users’ browser gets virtual host terminal, the user can do their experiment using Linux command. Such as Linux command ‘ls’, it will list files in the current directory.

Figure (3-6) Testbed Student Subsystem Gemini Interface
Figure (3-7) explains the steps taken by ShellInABox to get access to the virtual host assigned to the user group after authorization. ShellInABox will forward the user group identity information to the virtual machine and virtual host, and then returns the ShellInABox Ajax-terminal simulator to user’s browser. Gateway, virtual machine, and virtual hosts are Linux-based operating systems. In common, users will use Putty to get access to these systems through SSH. Users have to input group identity getting each of these Linux-based virtual machines. In this work, we use ShellInABox as Linux Command Server to manage all those processing. In this way, users release from tedious processing and focus on their experiment. Compared to Putty, the testbed front-end is a user-friendly web application.

Figure (3-7) Testbed Student Subsystem Virtual Host Access UML Sequence
In large scale system, testbed student subsystem will forward Linux access identity or Linux command from virtual machine to virtual host one by one through TCP/IP protocol. The testbed does not need to return the response message to users every time. When getting the server-side terminal emulator of the virtual host through ShellInABox, it will send response or result to user’s browser. The efficiency of putty is no better than the web-based system for the latency in the response to messages from each virtual host or virtual machine. Therefore, the web-based system takes more advantage than traditional systems using putty in large scale.

3.2 Administrator User Interface

In testbed administrator system, there are four main functionalities: view virtual/physical host, view virtual network, create a virtual network and delete the virtual network. We implement these functionalities using web-application technology including HTML, JavaScript, CSS, AJAX, jQuery and JSP. When the administrator clicks view button or hyperlink, the system will display the virtual machine and user group information in the database. Clicking hyperlink, the system will update Gateway database and virtual machine or virtual host setting.

3.2.1 Add a virtual network

Figure (3-12) shows the addition of virtual network function front end in the testbed admin system. Admin can upload a topology file for a virtual network to Apache Tomcat installed in Gateway. When clicking submit button, the Java Client program will check the topology and compute the resources including the number of hosts, CPU usage, and memory usage in each virtual machine through the MySQL database which keeps recording the status of each virtual machine resources. According to the recourses of each
virtual machine, Java Client program will send Linux command to create new virtual host and configure the IP address and network route set-up. In the virtual machine, it will create a user group and bind the group with the newly created virtual host. When all virtual hosts are set the GateWay, Java Client program will update the virtual machine related resources in MySql database. In this way, admin can get the virtual machine resources information and distribute Linux command to each target virtual machine more efficiency. The admin can also write the topology in the text area shown in the Figure 3-8. In the text area, we use three virtual hosts to create a topology. The router will create two virtual networks with IP address ‘10.10.1.101 24’ and ‘10.10.2.101 24’. The topology will use two virtual hosts which created a virtual network with IP address ‘10.10.1.101 24’ and ‘10.10.2.101 24’. There is a heap algorithm deciding which virtual host will be used.

![Image](image.png)

Figure (3-8) Testbed Admin Subsystem Add Virtual Network Front-end

Figure (3-9) is an example for creating a topology. In this topology, we need three virtual hosts M1, M2, and M3. The letter ‘H’ means host and ‘R’ means router. Therefore,
M1 and M3 are hosts. M2 is a router. M2 will use two virtual network cards. One’s IP is '10.10.1.100  24', the other’s IP is '10.10.2.100  24'. M1’s IP is ‘10.10.1.101  24’. M2’s IP is ‘10.10.2.101  24’. In Figure (3-10), it shows the topology in the system.

<table>
<thead>
<tr>
<th>HOST 1</th>
<th>HOST 2</th>
<th>HOST 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2</td>
<td>M1</td>
<td>M3</td>
</tr>
<tr>
<td>10.10.1.100</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>10.10.1.101</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>10.10.2.100</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>10.10.2.101</td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

Figure (3-9) Topology Commands

Figure (3-10) Topology Created by Command

### 3.2.2 View physical host

In admin subsystem, we can get all the physical hosts information of our system. It will display host details including the host name, the total number of NICs, IP address of each NIC, total memory size and used memory, history of used memory, CPU usage, and history CPU usage. In Figure (3-11), it displays the two hosts’ information.
Figure (3-11) DefaultGW and DMZ Virtual Host Information in Testbed

We achieve these tasks by implementing a Gateway Java client utility and a Java Server utility on each participating physical host. Admin sends a host information request via Gateway system. In Gateway, the Java client utility receives the request from admin and then forwards to physical host Server through TCP/IP protocol. The host, then, returns its information to the administrator via Gateway. The history of CPU and memory usage is stored in the MySql database installed in the Gateway system, and the admin can get history from it. Each host’s information is stored in the Gateway MySql database through the Java Client daemon on it. The interval to store host information can be set by Java client utility in the gateway. In this way, admin can get host information more effectively Figure (3-12).
3.2.3 View/Delete a virtual network

The administrator system can list all the virtual networks hosted on different hosts. The admin interface shows the virtual hosts information including the group owner, virtual network id, and host machine in which it resides for a virtual network. All those information are stored in the database installed in the Gateway. When setting the virtual network initially, virtual host and user group information get stored in the database.
In Figure (3-13), admin can delete a group virtual network only clicking the username hyperlink. We can get the virtual network information through MySQL database. In order to improve efficiency, we store virtual machines information in MySQL database and get the existing group name information, group virtual IP addresses, and virtual hosts’ name. When clicking the username hyperlink, the admin will send a deletion request to Gateway. The Gateway will forward the command to virtual machine such as DefaultGW. The virtual machine will delete virtual host LXC, remove the user group and updates the access configuration file. After this, Gateway will delete the virtual host information stored in MySQL database through Java Client shown in Figure (3-14).

Figure (3-14) Virtual/Physical Host Information Flow
Chapter 4

System performance and evaluation

4.1 System performance

The performance of our testbed is evaluated using V-NetLab [13] template virtual networks on an ESXi host in our previous work [21] and forms the basis for the current testbed as well. In the testbed, we assigned virtual hosts to a group of students shown in Figure (4-1).

![Course Assignment Network Topology](image_url)

Figure (4-1): Course Assignment Network Topology
There are some parameters need to be measured including virtual host network bandwidth, CPU, and physical host memory usage while running virtual hosts usage, and maximum path network latency in the topology when several virtual networks traffic co-exist.

To evaluate the network bandwidth, we use iperf [31] as a test tool in host1 and host4. We install iperf UDP client in host1 and server in host4. In Figure (4-2), we observed that the trend of bandwidth from host1 to host2 is decreasing with the increase of the number of virtual hosts.

Figure (4-2): Brand Width Per Host (Mbps)
To evaluate average CPU usage, we used sysstat/sar [32] as a test tool. In Figure (4-3), we observed that CPU usage is increasing with the number of virtual hosts.

![CPU Usage](image1.png)

**Figure (4-3): CPU Usage**

However, the CPU maximum usage is 13.5% showing the physical CPU is not over-utilized.

![Memory Usage](image2.png)

**Figure (4-4): Memory Usage**
To evaluate memory usage, we used sysstat/sar [25] as a test tool. In Figure (4-3), we observed that memory usage was increasing with the number of virtual hosts. The maximum usage is 35% for 20 virtual hosts. The minimum usage is 16% for two virtual hosts.

![Graph showing latency relation to number of virtual hosts](image)

Figure (4-5): Latency Relation to Number of Virtual Hosts

Host1 and Host3 are separated by three hops. The average latency in the network traffic is measured on the Host1 to the Host3 path. In Figure (4-4), we observed that the trend of latency increases from 2 hosts to 20 hosts. The maximum latency is 6.8ms for 20 virtual hosts which are not significant for academic experiments.

### 4.2 Discussion

In class, there are three tasks learning network security knowledge through our system.

Lab 1 is Network and Service Configuration task. Students need to refer to the tutorial to access the testbed. After they successfully start virtual networks, they need to run commands such as ping, ifconfig, route inside each virtual host to show the network
information. There are two phases in this lab assignment. In part 1, their task is to figure out the static topology of the given network and verify whether it is fully connected, that is, within a network, using ping from any computer to another one. In part 2, students need to set up and start a few services including Web Server and SSH Server. When setting up these services, they need to pick any virtual hosts to be the servers for them. In the report, they need to draw a picture to illustrate the static topology of your network. Information such as hostnames, IP addresses, and sub networks need to be included in the topology. They need also label each host that runs a particular service, for instance, the web service is running on DMZ. Also, describe the configuration parameters for each service. Including a paragraph summarizes the task that has been done by each member in your group.

Lab 2 continues suing the lab environment from the previous lab assignment. In part 1, students will prepare network with a combination of services and develop scripts to generate network traffic that makes use of the services. After that, they need to configure the network in the way that only one host Infrw (as it will allow interception of inter-subnet network traffic) is accessible to another group. But students need to make sure that it’s possible to figure out the service locations as well as dynamic traffic in network through the access of that single host. Then they need to hand over your network to another group, which is responsible for investigating and discovering network. In order to create dynamic traffic, you can first write a bash script. In part 2, students will be assigned the virtual host log-in information of another group for the discovery of network services running in its virtual network. During that period, the original group should not make configuration changes to their network. They need to write two reports consisting
of descriptions for their network and the assigned to-be-discovered network. For the assigned to-be-discovered network, they should evaluate how difficult to discover, describe discovery results, including the service locations as well as all the observed network traffic, and how to discover and analyze each result.

In lab 3, students need to write and invoke iptables rules on Intfw (and/or Extfw), to get the following tasks done. Run iptables, Block ping (ICMP) packets, Block WWW traffic, Block SSH traffic and so on. They need to write a report summarizing solution to each of the tasks above. Describe the task of each member in your group. After those tasks, we took a survey of students to evaluate our testbed.

In Figure (4-6), we took a survey of students using a scale of 1 to 10 (1 is lowest, 10 is highest). From survey sample number 1 to 7, they are represented the belowing questions. No 1, How do you like the virtual network environment in general? No 2, Did it help your learning experience in this network security course? No 3, Did you like the features in testbed? No 4, What are your comments in lab1? No 5, What are your comments in lab2? No 6, What are your comments in lab1? No 7, Would like to work on this testbed if we have more time in the course? We can see most of the students marked this system more than 5
Although the survey is good, the testbed should be more customizable. Once the topology was submitted for the network, it would be nice to see a virtual implementation of it. After login, the system should remind the team what they needed to protect and change. A small list of commands to refresh Unix knowledge would’ve been good. It would be better to see it as a fully GUI based system rather than only terminal based. The system should provide an opportunity to mess with a setting and change things without affecting an actual network. The lab instructions were easy to understand and implement. Student liked being able to log in from anywhere. Students liked the lab environment overall.

Working with VMs in a virtual network is a really great way to try out some of these concepts. For lab one, the instructions were clear and easy to understand, and the lab went smoothly. For lab two, students liked trying to hide and find certain things to trick other groups. Students think that this lab seemed to be the hardest to do since one would
need to be advanced understanding of Ubuntu to completely conceal information to others. For lab three, students think this lab is fairly straightforward after understanding the second lab, but the material to learn about iptables was a little bit limited. Students think given more time or direction they would've been able to pick it up more quickly. It is better to provide students more iptables information and practice.
Chapter 5

Conclusion and Future work

In this thesis, we provide a virtualized environment for students or researchers performing assignments and experiments. We will discuss a few important features in the testbed. First, our testbed based on open-source Linux operating system. There are many distributions available and customizable providing a close interaction with physical host resources. It is a good option for researching and teaching. Second, LXC based virtual host share virtual machine kernel and isolate each virtual host. It can prevent virtual machine and physical host from malicious access. Even a hacker can get access to the virtual host, the physical host, and the virtual machine cannot be harm. LXC can support virtual network for each virtual host using the same source on virtual machine assigned from physical host. This network infrastructure can provide a more flexible in a scalable system.

Finally, our testbed offers an automatic topology generator with a user-friendly interface. Instructors can realize the network topology, user group identify, and IP address with only clicking a button. The flexibility, scalability, and maintain are very convinced.

In this thesis, we proposed a light-weight, flexible, and web-based testbed for computer & network security experiment. The testbed provides a flexible and automatic
creating a virtual network topology over a limited resources physical machine or virtual machine. It also makes the system more scalable and maintainable. We use a V-NetLab architecture tool to evaluate testbed performance. The virtual host networks used quite a reasonable usage of resources. It can save more virtual host resources than the physical host. We design experiments for students who had V-NetLab experience. They perform the same network setting and security tasks such as creating the topology, seeking network service and network configuration and monitoring network traffic. They can accomplish their tasks very fast through user-friendly testbed interface.

We found that the testbed virtual host terminals are not very stable. We plan to create a more stable terminal service instead of ShellInAbox. When creating a complex topology, the automatic creating topology function is time-consuming. We plan to optimize topology creating an algorithm to improve the efficiency. We plan to use this system for more network classes and design more labs for students to evaluate our system. Also, we will let more instructors and researchers use this system for their class by providing detailed manufactory. After getting feedback from those users, we will improve our system further.
References


[20]. Interdisciplinary capstone [Online]: Available: 
http://seniordesign.engr.uidaho.edu/index.html


http://oit.duke.edu/comp-print/labs/vcl/index.php


