CLaaS: Cybersecurity Lab as a Service

Cihan Tunc* and Salim Hariri
The University of Arizona, Tucson, Arizona 85721 USA
{cihantunc, hariri}@email.arizona.edu

Abstract

The explosive growth of IT infrastructures, cloud systems, and Internet of Things (IoT) have resulted in complex systems that are extremely difficult to secure and protect against cyberattacks which are growing exponentially in complexity and also in number. Overcoming the cybersecurity challenges is even more complicated due to the lack of training and widely available cybersecurity environments to experiment with and evaluate new cybersecurity detection and protection methods. Therefore, the goal of our research is to address the education, training, and experimentation challenges of the cybersecurity by exploiting cloud services. In this paper, we present the design, analysis, and evaluation of a cloud service, that we refer to as Cybersecurity Lab as a Service (CLaaS), which offers virtual cybersecurity experiments that can be accessed from anywhere and from any device (desktop, laptop, tablet, or mobile device) with Internet connectivity. In CLaaS, we exploit cloud computing systems and virtualization technologies to provide virtual cybersecurity experiments and hands-on experiences on how vulnerabilities are exploited to launch cyberattacks, how they can be removed, and how cyber resources and services can be hardened or better protected. We also present our experimental results and evaluation of CLaaS virtual cybersecurity experiments that have been used by graduate students taking our cybersecurity class as well as by high school students participating in GenCyber camps.

Keywords: CLaaS, virtualization, cybersecurity experiments, virtual cloud services, cloud computing, education

1 Introduction

The explosive growth of IT infrastructures, cloud systems, and Internet of Things (IoT) have resulted in complex cyber systems that are extremely difficult to secure and protect due to many factors such as their size, architecture complexity, distributed nature, heterogeneity, the large numbers of users, and diversity of services provided, just to name a few. These challenges, coupled with a shortage of skilled cybersecurity experts and the extreme difficulty in setting up cybersecurity experimental environments, have resulted in vulnerable cyber systems with no adequate technical support. Therefore, there is a critical need for cybersecurity testbeds to test and evaluate new cybersecurity algorithms and techniques. Furthermore, there is also a critical need for cybersecurity education and training environments that can be accessed from anywhere, any time, and by any device to study the vulnerabilities of the existing methods without using complex environments.

In this paper, we present the design and the analysis of Cybersecurity Lab as a Service (CLaaS) that offers virtual cybersecurity experiments as a cloud service that can be accessed from anywhere and from any device (desktop, laptop, tablet, smart mobile device, etc.) with Internet connectivity. The CLaaS enables users to conduct virtual cybersecurity experiments in a closed virtual cloud environment to: a) understand the methods and techniques that are used to launch cyberattacks; b) provide hands on and experimentation with existing cybersecurity tools (e.g SNORT, Wireshark, etc.); c) learn how to detect

Journal of Internet Services and Information Security (JISIS), volume: 5, number: 4 (November 2015), pp. 41-59
*Corresponding author: NSF Cloud and Autonomic Computing Center, The University of Arizona, Tucson, Arizona 85721 USA, Tel: +1-520-262-6345

vulnerabilities in networks, operating systems, and applications and also how to perform penetration testing; and d) evaluate new cybersecurity detection and protection algorithms.

Current approaches for cybersecurity education and training involve setting of a complex cybersecurity laboratory that consists of many dedicated computers, network interface cards, routers, switches, and specialized cyber security software tools [32]. Hence, setting and configuring an experimental cybersecurity environment is a very time consuming task and it requires deep knowledge in software systems, hardware, and network devices so that any misconfigurations, misbehaviors, etc. will not cause a failure in the other connected systems. On the other hand, the required expertise and knowledge are available only in few elite schools and not reachable worldwide. In contrast to these methods, the CLaaS offers complex and sophisticated virtual cybersecurity experimental environments in a transparent manner; the users just focus on the cybersecurity issues being investigated or studied in the experiment using a web-browser so that they can access anywhere and anytime and do not spend their time with building a cybersecurity testbed. Hence, while the cybersecurity environment is controlled in a more secure way, required operational cost, space, and time are minimized [29]. To achieve the desired transparency, we exploit cloud computing systems and virtualization technologies. We present our evaluation to CLaaS virtual cybersecurity experiments that have been used by graduate students taking our cybersecurity class as well as by high school students participating in GenCyber camps that were sponsored by NSF and were held in summer of 2014 and 2015.

The rest of the paper is organized as follows. In Section 2 we give a brief overview of related work. Section 3 describes the CLaaS architecture and design. In Section 4, we present the virtual cybersecurity experiments and the testbed used to deploy those experiments. We discuss our experimental evaluation results in Section 5. In Section 6, we conclude the paper and discuss our future research activities.

2 Related Work

Cybersecurity classes and experiments have been used where students either use dedicated physical environments built a-priori or asked to build their own cybersecurity environments (e.g., using VirtualBox [26] or VMware [14]). In both cases, it is extremely difficult to make these laboratories widely available to students, except for a few universities or schools. When the dedicated environments are used, the students will be using the privileged admin account to be able to run the required cybersecurity tools which can cause systems failure due to mistakes or misconfigurations. In addition, this usage can result in illegal attacks to the other systems or networks that might lead to large network failures. Furthermore, when students are asked to create their own environment using virtualization, to reduce the burden of setting the dedicated laboratories, configurations take a long time, misconfigurations occur, and, as a result, the students spend their times mostly on configuring their own testbed instead of spending their time experimenting with cybersecurity detection and protection techniques. Therefore it is critically important to develop a cloud service and ready-to-use cybersecurity training programs that focus on teaching and training with cybersecurity algorithms, rather than building everything from scratch by students and for every experiment [31], [32].

Providing such an environment requires using virtualization techniques. Currently, there are several different virtualization technologies that can be used to create virtual resources:

1. Hardware Emulation: This technique creates a virtual machine to emulate the hardware on the host system as in QEMU [11] and Bochs [3].

2. Full Virtualization: This technique uses a hypervisor that allows sharing host system hardware and requires instruction trapping and handling [22] as in VMware [14] and KVM [8].

3. Para-virtualization: In this approach, the hypervisors do not trap or handle instructions, but instead the guest OS of the VM is modified on the kernel level to allow this virtualization and the host system hardware is shared by the hypervisor [27]. The most commonly used example is Xen [30].

3 Virtual Experiment Environment Development

In this section, we discuss our approach to develop the virtual components required to implement any cybersecurity experiment, such as virtual network interface and network components.

3.1 Virtual Network Interface

Virtual Network Interface (VNI) is an abstract network interface that does not necessarily correspond to a physical network interface. In most of the new operating systems, the network interface (from a kernel perspective) is nothing more than a software object that can process network packets. And, the process of transmitting or receiving packets is left for the interface driver. Thus, all operating systems that support virtual network interfaces maintain a table of these interfaces. Such operating systems usually allow software to interact with VNIs [7].

3.2 Software Network Switch

A switch can operate at different levels of the network hierarchy. For example, a bridge can be viewed as a switch to connect two network segments together to form a new network segment and it operates at layer 2 protocols. Modern operating systems allow the creation of a software bridge that connects physical network interfaces as well as VNIs [4] (e.g. bridge-utilities in Linux systems).

3.3 Virtual Network

Many modern operating systems and hypervisors (e.g., Xen, KVM, and VMware) support creation of the virtual network interfaces and bridges. For example, in Figure 1, the network consists of a router with three separated networks that connect five hosts with one firewall, three switches and two servers. Figure 2 shows the connections of NICs to form the network topology shown in Figure 1.

When virtualizing the whole network, the physical switches are represented by the bridges, while the VMs take the role of routers, firewalls, hosts, and servers as shown in Figure 3.

Another advantage of virtualized networks is that they could be combined with a physical network. For example, Figure 4 shows a cybersecurity experiment topology (i.e. Intrusion Detection System) that consists of three machines: one of the machines is used to generate normal traffic, the second one is the attacker machine, and the last machine is the virtual network environment. Figure 5 presents the hybrid environment.

3.4 CLaaS Development Approach

Figure 6 shows our implementation approach of CLaaS on a private cloud system that utilizes multiple high performance Intel servers.

In our approach, the user logins to the CLaaS website using a web browser and selects a pre-existing cybersecurity experiment that was built using VMs and private virtual networks as discussed previously. After the login, the user connects to the CLaaS Front End Web Server which provides the existing virtual cybersecurity experiments or labs as shown in Figure 7.

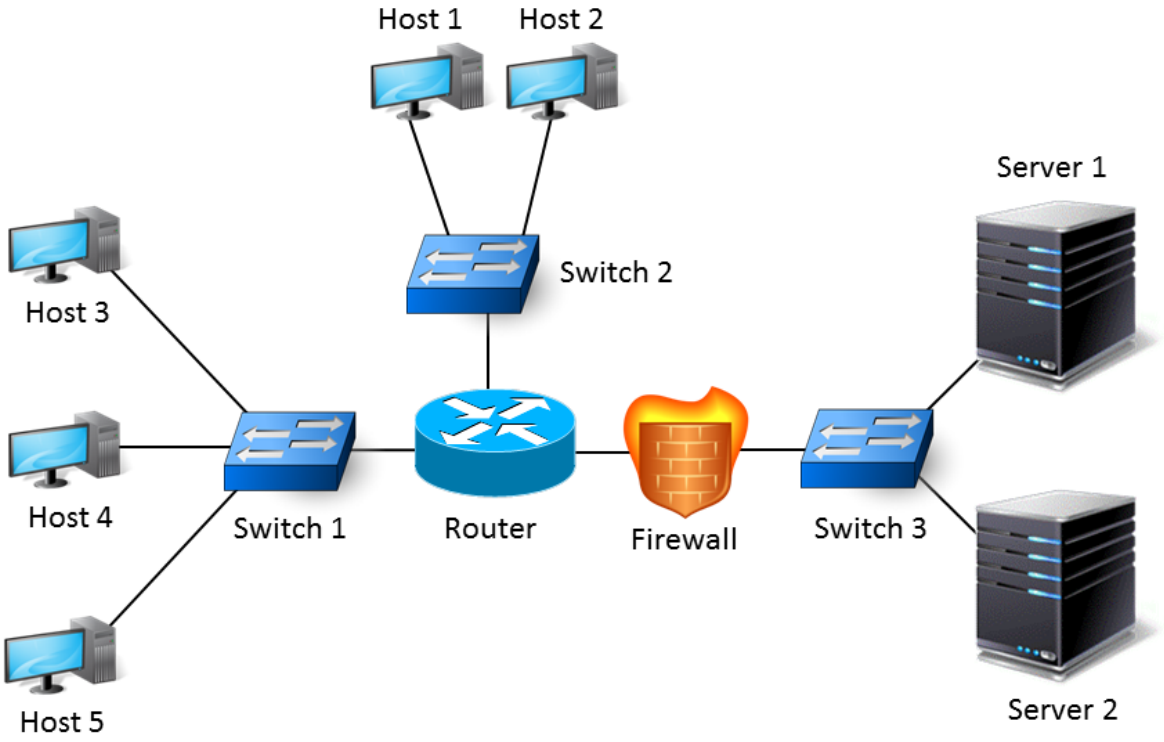

Figure 1: Example network topology.

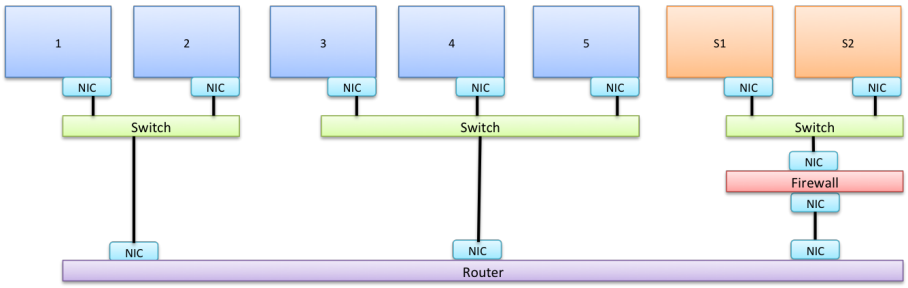

Figure 2: Example network NIC connectivity map.

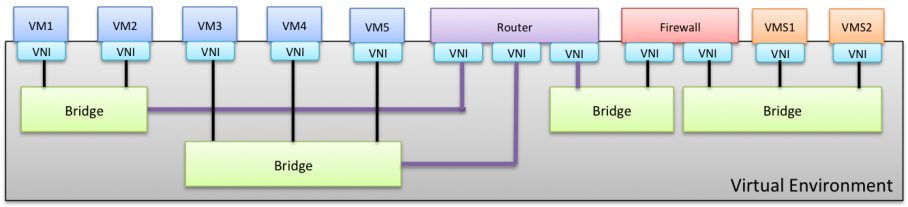

Figure 3: Example virtualized network

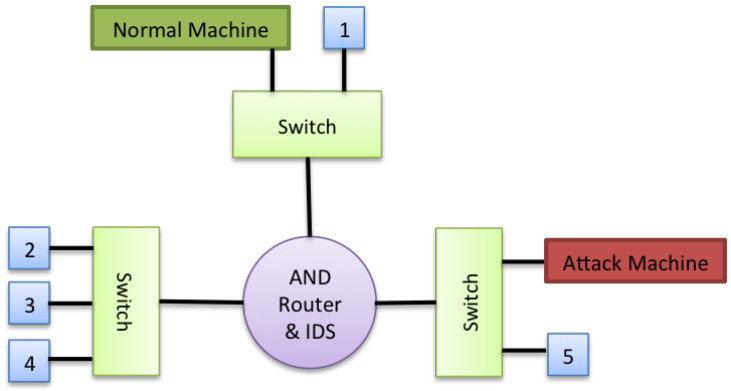

Figure 4: Intrusion Detection System Testbed topology

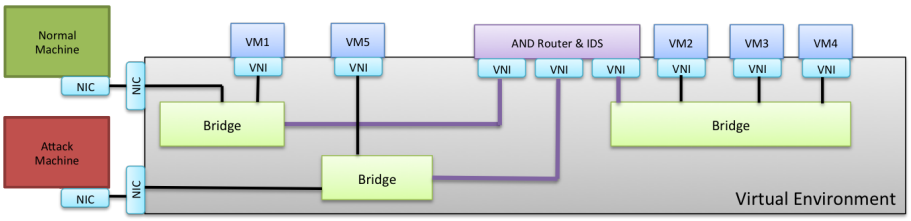

Figure 5: A hybrid cybersecurity experiment environment

The CLaaS user interface runs on Apache and uses Postgres database that stores all user profile information (e.g. username, password) as well as the detailed information on the virtual cybersecurity experiments. To be able to provide a GUI to the virtual machines (VMs), VNC is installed to enable users to access CLaaS experiments from any device (computer, tablet, even a smartphone), with Internet connectivity. This feature allows users to manage the VMs as if they are using their local computers (with the additional capabilities of rebooting).

The CLaaS backend consists of a Controller, a Resource Pool, and private cloud with cloud infrastructure services. The CLaaS Controller manages the whole environment, such as creating a new set of experiments, creating virtual switches, assigning IPs to the VMs, or starting/stopping/deleting the VMs using VMware ESXI API as well as VMware vCenter. It also decides which host machine to be used to create the virtual cybersecurity experiment based on the available resources on these systems. The CLaaS has a local resource pool that includes four repositories: 1) VM configuration repository (Configuration Repo); 2) OSs templates repository (OS Repo); 3) Software repository (Software Repo); and 4) Environment repository (Environment Repo). VM configuration repository stores VM configuration templates to determine the number of cores, assigned main memory, and number of network interfaces. The OS repository will have the images of the OSs that are used by the environment and their metadata; for example the minimum core, minimum disk space, and minimum memory requirements for Ubuntu, Debian, Windows, etc. The software repository stores the software packages (e.g. Wireshark, network attack tools), configurations of the packages, and deployment conditions. The environment repository stores information about the whole test environment or an experiment, i.e., it stores what type of VM configuration required, what OSs images to be used on what VMs, how the OSs are configured, what software to run and where, what configurations are needed for the software, and finally what is the network structure that connects the environment.

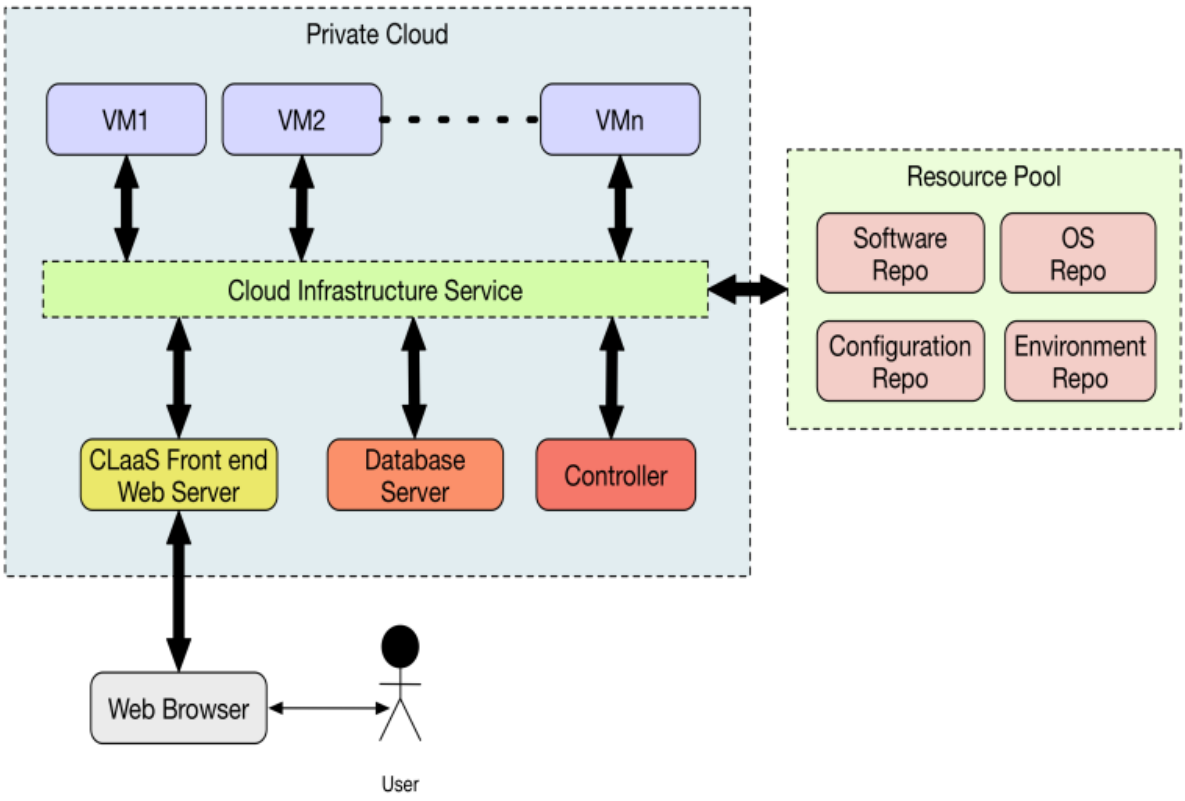

Figure 6: CLaaS implementation on our private cloud

Using the cloud infrastructure and the resource pool, the controller builds the cybersecurity experiments and the algorithm to create a new experiment is shown in Algorithm 1. Step 1 gets the username, the list of available host machines, experiment id, and the list of the VMs that can run the selected experiment. Among the available host machines, the host machine with the most available resources is chosen to reduce possible performance bottlenecks (Step 2). Next, a virtual switch is created on the selected host machine for the user and experiment id (Step 3). The new VMs are linked based on the templates (Step 4) so that the cloning time will be minimized as well as the hard drive space usage. Each VM is connected to two virtual switches: The first virtual switch is for the connection of the VMs to the Internet whereas the second virtual switch is used for the internal VM communications required to run the selected experiment. This allows users to apply any kind of attacks without affecting other systems. In Step 6, the IP addresses are configured – DHCP is used for the Internet connection and a static IP list is used for the internal connection (private switch). The IP addresses are verified (Step 7) and send back to the database server and web interface for the access.

4 Experiments

4.1 Experimental Testbed

We have implemented CLaaS using a private cloud that contains five Intel high-performance servers (S2600GZ) with Intel Xeon E5-2680 (2.80GHz) CPUs providing 40 cores and 64GB RAM, as shown in

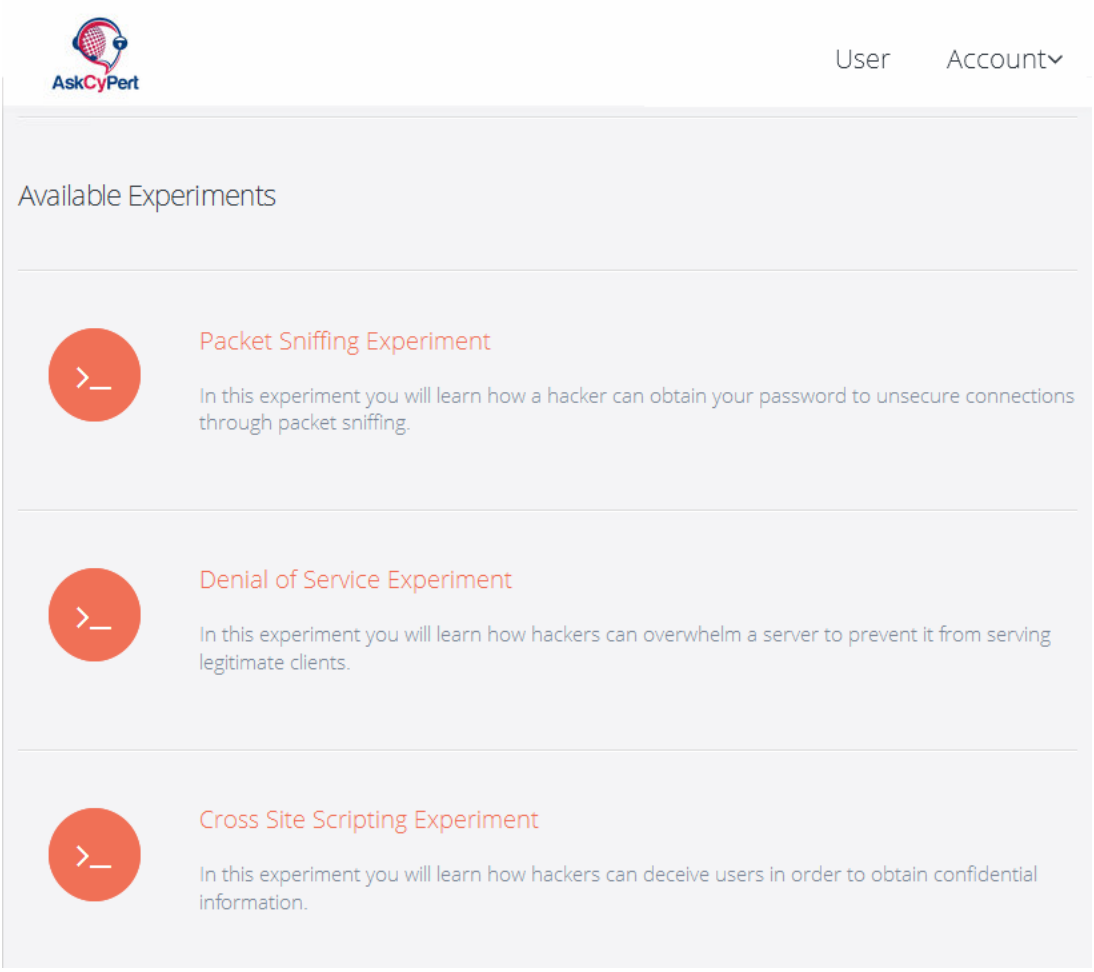

Figure 7: CLaaS interface for the available experiments webpage

Algorithm 1: The algorithm of creating a new virtual experiment

```
get the user_name, host_machines, experiment_id, required_VMs ;
which_host = load_balancing(host_machines) ;
vSwitch = create_new_vSwitch(which_host, user_name, experiment_ id) ;
new_VMs = creates_new_VMs(which_host, experiment_ id, required_VMs) ;
connect_VMs_to_vSwitch(which_host, new_VMs) ;
configure_VM_IPs(new_VMs, vSwitch) ;
get_VM_IPs(new_VMs) ;
```

Figure 8. The physical host systems are connected to a router with the capabilities of detecting misbehaviors and blocking their IPs/MACs [16], [19]. For the virtualization, we have chosen VMware vSphere Hypervisor (ESXi 5.5) due to its API and capabilities such as creating a virtual switch, linked-cloning VMs, and complete control/management of the environment. Also for the management function, we used VMware vCenter (especially for cloning of the VMs).

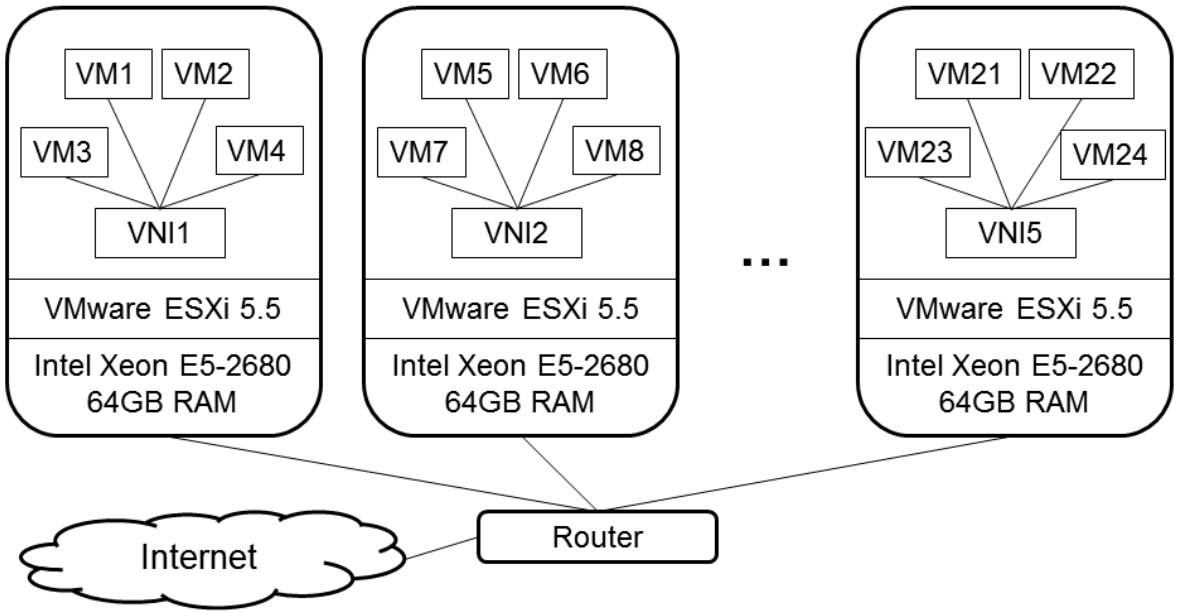

Figure 8: CLaaS implementation environment

4.2 Cybersecurity Experiment Scenerios

We have developed the following virtual cybersecurity experiments: (1) Brute-Force Password Cracking Experiment; (2) Packet Sniffing Experiment; (3) DDoS Attacks Experiment; (4) DNS Attack Experiment; (5) Buffer Overflow Experiment; (6) Cross Site Scripting (XSS) Experiment. In what follows, we describe each experiment setup and how the students are guided for understanding the cybersecurity issues, in further detail.

4.2.1 Virtual Brute-Force Password Cracking Experiment

Given enough computing power and time, an attacker can try all possible combinations or a set of possible username and passwords to gain access to a system, referred as brute-force attacks and dictionary attacks as one of the most commonly used technique to crack user accounts. And it is well known that the passwords that are either too short or too simple (even though they are longer) can be cracked using such techniques [21]. It has been also shown that the users generally prefer to use the simplest or weakest passwords allowed [20]. CBS News presented the top 25 worst passwords of 2014, revealing the use of very weak passwords such as “123456”, “password”, “qwerty”, “letmein”, “abc123”, “batman”, and their variances. Such a choice allows attacker to find the correct password by trying different possible permutations or using the password which is identical to the username, such as “test” - “test”. Therefore, the main attack scenario is still brute-force attack against remote services such as SSH, FTP, etc. [1]. Recent examples reported how hackers hacked passwords using brute-force tactics and third-party applications to access Apple user’s online data storage, leading to the subsequent posting of celebrities’ private photos online [24]. As a result, in order to emphasize the password security, we introduced students the brute force and the dictionary attacks and how they are executed. We have used two VMs (one as the victim and another one as an attacker) with Ubuntu 12.04 Desktop where they are connected to each other using a virtual switch for the password attacks. The students used Hydra [12] to figure out the

user password and experimented with different password sizes and combinations.

4.2.2 Virtual Network Packet Sniffing Experiment

The purpose of the network packet sniffing is the eavesdropping and understanding the packet contents and headers of communication protocols using Wireshark (a commonly used open-source packet analyzer [15]). This virtual experiment contains three VMs connected to a virtual switch as shown in the Figure 9 where the sniffer listens to the communication between the client and the server using Wireshark. The server VM operates as Telnet, SSH, HTTP, and DNS server. In order to evaluate this experiment, a basic web server is already installed in the server VM and the students are asked to understand how TCP and UDP and DNS protocols are identified and what other features Wireshark provides, such as packet filtering, packet structure analysis, etc.

In order to evaluate this experiment, a basic web server is already installed in the server VM and the students are asked to understand how TCP and UDP and DNS protocols are identified and what other features Wireshark provides, such as packet filtering, packet structure analysis, etc.

In this experiment, the students first start with the Telnet communication protocol where the communication between the server and the clients are not encrypted. The students are asked to analyze every package between the client and the server and identify the username and the password as well the executed commands.

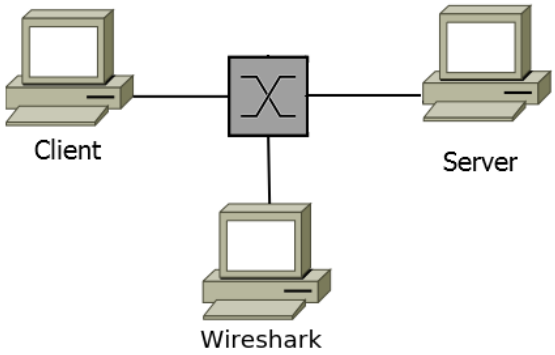

Figure 9: Network Packet Sniffing experiment scenario

While Telnet does not encrypt the communication channel, on the other hand, SSH uses a secure channel. Therefore, when SSH protocol is used the packet analysis does not provide any useful data without decryption. Hence, in the next step, the students are asked to use SSH protocol for the communication and try to find out the username and the password or any useful data, which is not possible.

After studying Wireshark to find out the username and the password of Telnet and SSH connections, the students are asked to use different protocols (e.g., DNS, HTTP, UDP) and asked to analyze their packets, headers, payloads, etc. In addition, the students are also asked to exercise multiple layers of encapsulation that go into creating network packets.

4.2.3 Virtual DDoS Attacks Experiment

Distributed Denial-of-service (DDoS) attack is one of the most common attacks targeting a server or a network resource such that it become unavailable to its intended normal users [18]. Based on a DDoS impact study from Incapsula [25], DDoS can cost upto $40,000 per hour and the average DDoS cost

can be assessed at about $500,000 affecting both IT and units such as security and risk management, customer service, and sales, and marketing, etc.

To study the effects of DDoS and to develop methods to prevent them, the DDoS attack needs to be well understood. Hence in this experiment, we provide an environment with a victim server VM, a client VM, and multiple attacker VMs as shown in Figure 10. On the server VM, a basic web interface is built using Apache web server and the client connects to the web server through a web browser. The attacker VMs have multiple common DoS and DDoS attacks to slow down and consequently crash the server network. It should be noted that all the VMs have two NICs (one is to provide Internet access while the other one is to provide internal communication for the virtual lab so that the attacks can be successfully tested without affecting other systems).

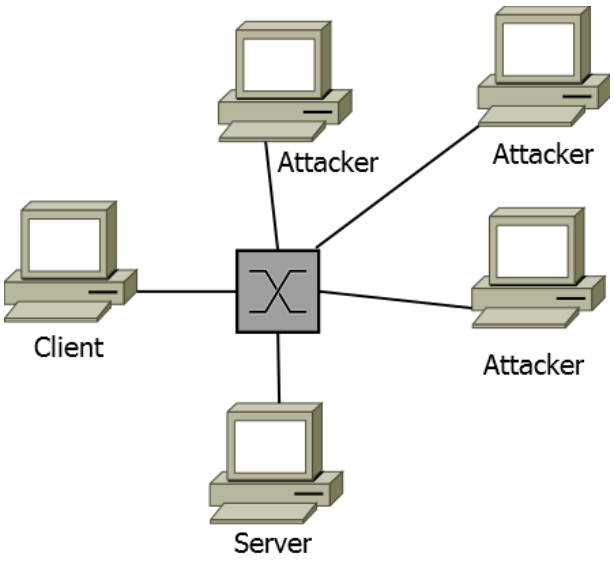

Figure 10: DDoS attack scenerio with 1 client, 1 server, and 3 attacker VMs.

The attacker VMs have NMap [9] and HPing3 [28] to apply common TCP attacks.

Nmap ("Network Mapper") a network discovery and security auditing tool which uses raw IP packets to determine hosts on the network and the operating systems and that are running on each host, etc. In addition, it can also be used to apply Slowloris DoS attack, IPv6 Router Advertisements (RA) Flooding attack, SMB flooding attack, etc. [10]. HPing is a network tool able to send custom TCP/IP packets to a network host for testing firewalls, port scanning, etc. [28], and it can be used to apply TCP SYN attack to a specific IP and port (e.g. sudo hping3 -i eth1 --flood -S -p 80 SERVER_IP) resulting in overload in the web server network. Similarly, we have provided Sockstress tool (an attack tool that exploits vulnerabilities in the TCP stack [23]) to apply TCP window size attack. Using these attacks, the students can experiment with DoS/DDoS attacks against a webserver. During the attacks, the students are asked also to monitor the network and the connections using Wireshark on the server VM. Next, the students are asked to search the literature to find out possible DDoS attacks affecting the systems and how the systems have been secured against these attacks.

4.2.4 Virtual DNS Attack Experiment

The Domain Name System (DNS) is a hierarchical distributed naming system for the resources connected to the Internet and/or a private network translating domain names to the IP addresses. However, DNS

protocol is vulnerable to attacks since DNS protocol was not designed initially to handle security [17]. In this virtual cybersecurity experiment, we show how DNS works, what the insecurities of the DNS are, and discuss how it can be protected. This experiment is composed of five VMs connected by a virtual Ethernet network (in addition, they are connected to the Internet via the standard switch). In this virtual cybersecurity experiment, one VM is used as a client VM and another VM is used as an attacker VM. The students are provided a DNS server and two additional VMs (as DNS server #2 and #3) to be used for the cache poisoning attacks.

First, the students are expected to understand how DNS protocol works to resolve domain names to IP addresses using the Wireshark installed on the server VM. Next, the DNS security flaws to be exploited are discussed. Specifically, in this virtual experiment, the students learn how the DNS cache poisoning can be carried out by spoofing DNS responses (that arrives before the actual DNS response) and DNS rebinding as the two main DNS vulnerabilities. Then, several security techniques are discussed to protect DNS protocol operations such as DNSSEC [6]. For the DNS server, Bind9 is used since it is the most popular DNS server software used in IT environment [2]. Furthermore command line dig tool (domain information groper) is used to query the DNS name servers [5]. The following tasks are performed in the DNS cybersecurity experiment:

- Task 1: DNS poisoning

  In the DNS poisoning attack, wrong data is introduced to the DNS resolver's cache so that the DNS will be returning an incorrect IP address to the victim. To demonstrate this attack, the following steps are carried out by the students:

  - Step 1: First, the students are asked to update */etc/resolv.conf* file with the local DNS server IP instead of public DNS servers (the DNS information is retrieved from the local DNS server that obtains the information from Google DNS service (8.8.8.8)). And, then, the DNS servers are tested using dig command to various webpages and also Wireshark.
  - Step 2: The students are given a DNS hijacker program which listens for the DNS packets and spoofs responses for them. In the second step, the DNS-attacker VM is used to apply DNShijacker script that listens for DNS packets and spoofs the responses. This program will listen for DNS packets and spoof responses to them
  - Step 3: Using this DNS hijacking tool, the students poison the DNS cache by showing a wrong DNS server and the dig query returns a different IP address.

  It should be noted that the spoofed response will only be accepted by the client if it arrives at the client before the actual response from the server. Hence, the cache poisoning method used in this experiment targets the client directly; however, most cache poisoning attacks target DNS servers.

- Task 2: DNS Rebinding

  In the second DNS attack type, DNS Rebinding, the attacker uses a malicious web page to control the victim's network-level so that the attacker can create sessions in web servers. The steps of this experiment are discussed as follows:

  - Step 1: On the DNS client VM, using the web browser, http://example.com/rebind.php is visited that replies with a very small TTL so that it will not be cached.
  - Step 2: On the DNS attacker VM, the users update the /etc/hosts file and restart the dnsmasq service to load the updated configuration into the DNS server.

  - Step 3: On the client VM, after launching the malicious software given by the webpage, the server IP displayed changes which shows that the connection has been hijacked using DNS rebinding.

  Our rebinding attack required manual changes to the configuration of the DNS server. Most real-world rebinding attacks would use automated methods.

Some of the attacks that we exploited, including the cache poisoning attack, can be prevented using DNSSEC and similar protections to ensure the authenticity of the transferred zones. After the experiment tasks are fully executed, students had the opportunity to learn how DNS protocol is attacked, and how to detect and protect this protocol's operations, without the need to setup any lab, they just focus on the steps to explain the protocol vulnerabilities and how they are exploited by attackers.

4.2.5 Virtual Buffer Overflow Experiment

The purpose of this experiment is to demonstrate the importance of buffer bounds to prevent the execution of arbitrary code by injecting code (i.e. shellcode) that launches a Linux shell into a buffer. This experiment first overflows the stack and then overflows the heap using one attacker VM and one victim VM.

The students use GNU Debugger for analyzing vulnerable programs and smashing the stack and the heap. The vulnerable example program is written to take input from the user as a command line argument. Then, it uses memcpy() to copy the argument to a string buffer. However, because it does not check the length of the argument, if the argument is larger than the size of the buffer, memcpy() will write to an area of memory larger than the buffer.

It is important to note that the vulnerabilities exploited in this experiment have been mainly blocked in the modern operating systems (especially in recent versions of Linux). Normally, using the NX (no execute) bit set for the stack data segments, CPU is prevented from executing commands in those segments. However, for the demonstation purposes, we have disabled this feature.

- Task 1: Stack Overflow

  The vulnerable example program takes input from the user as a command line argument. Then, it uses memcpy() to copy the argument to a string buffer without checking the length of the argument. As a result, if the argument is larger than the size of the buffer, memcpy() will write to an area of memory larger than the buffer, resulting in a stack overflow. Using this approach, the students are guided to execute an external code to gain access to the shell.

- Task 2: Heap Overflow

  Similarly, in this task the students exploited heap allocation and tried to run a different function. Since the heap is dynamic, the students used gdb to calculate the address and the distance of the pointer and the memory. Finally, the students were able to run an external function.

4.2.6 Virtual Cross Site Scripting (XSS) Experiment

Cross-Site Scripting is the most common application layer hacking technique [4] where an attacker exploits vulnerability in a website's coding in order to have it execute external scripts on a victim's machine to steal web sessions and data. If an attacker gets a hold of the victim's authentication cookie and stores it in his or her machine, he or she can then go to the target website and the website will read the cookie and assume that the victim is the one accessing the site. This enables the attackers to access all the victims' information.

In this virtual experiment, the students are walked through a simple cross site scripting example using one attacker machine stealing cookies using a cookie stealing software from a session and a victim machine logging into a simple webserver with a database connection.

5 CLaaS Performance Evaluation

During the deployment of the virtual cybersecurity experiments, we first create template VMs to be cloned for the users and the experiments on the host machines. There are two types of cloning: full clone (copies the VM completely including the full copy of the virtual hard drive and updates the virtual resources such as the MAC address) and linked-clone (creates only a delta disk, instead of the complete disk, that only has the change information; therefore delta disk is much smaller than the complete virtual hard drive of the VM in size). Since the linked cloning creates only a delta disk, the time required to create a VM clone decreases significantly. For example, a full clone in our system takes 4:15 min whereas a linked clone takes only 50 seconds. This also reduces the required communication bandwidth and also the system hard drive storage requirements.

In Figure 11, we show the amount of time it requires to create a new virtual cybersecurity experiment with respect to the number of VMs required. In our environment, during the creation of a new experiment, the main bottlenecks has been the time required to create a linked-clone and the time needed for the VMware Tools to start correctly (VMware Tools are used for the remote management of the VMs and obtaining their environment information, such as the dynamic IP).

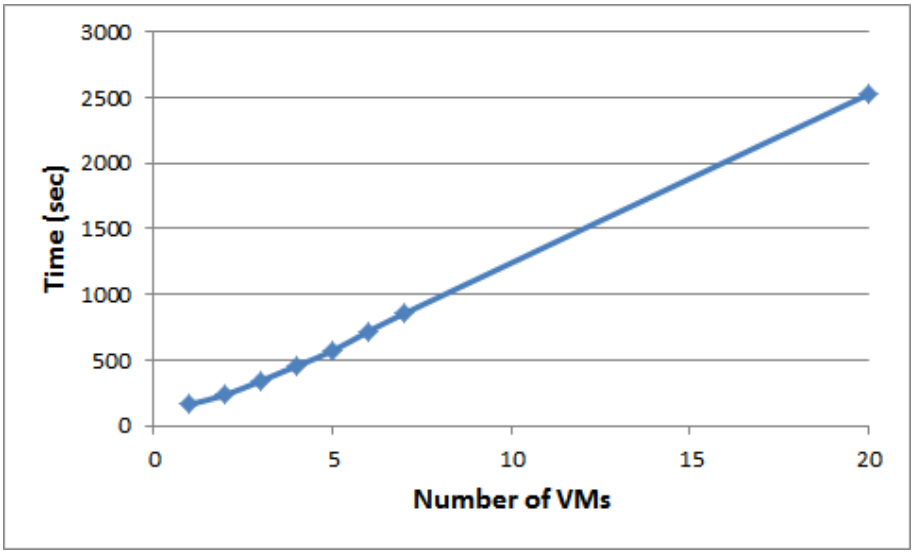

Figure 11: The time (in seconds) required to create a virtual cybersecurity experiment with respect to the number of VMs.

We have also monitored the resource utilization of the host systems during the creation of the cybersecurity experiments. Figure 12 shows the CPU utilization during the deployment operations with upto 17.5% due to the computing operations during cloning and configuration of the VMs (including the reboots). Similarly 13 shows the disk latency upto 17.5ms. Figure 14 shows that the deployment of the experiments did not affect the network bandwidth since the VM templates have been copied to each host machine to reduce both the network and the disk operations. Finally, Figure 15 shows the amount of allocated memory while the number of VMs deployed for the virtual experiments increases. The allocated

memory reaches close to 100% due to the reservation for the VMs, even though no memory related activities are performed. In our environment, each server has 64GB RAM and with the deployment of the 64 VMs (each of them has 1GB virtual memory), our system becomes completely utilized, and, hence the memory acts as the main bottleneck for creating further VMs.

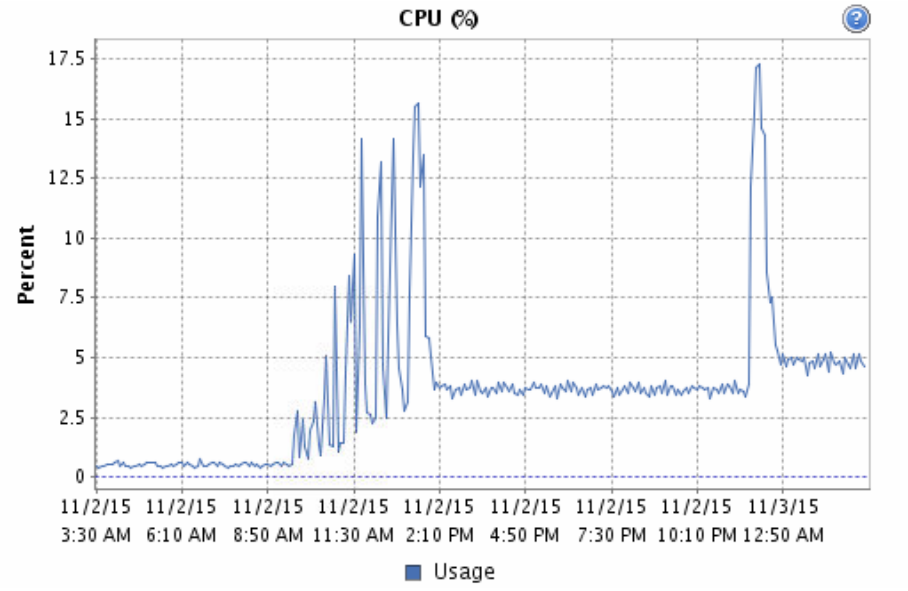

Figure 12: CPU utilization during the large deployment.

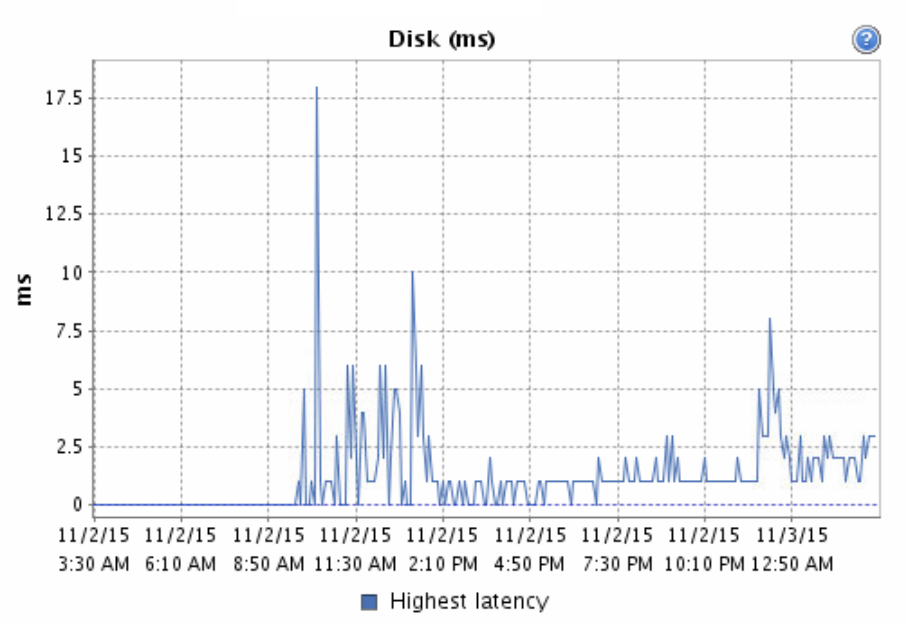

Figure 13: Disk latency (in ms) during the large deployment.

We have monitored the CPU and memory utilization of the host machines and observed that, during

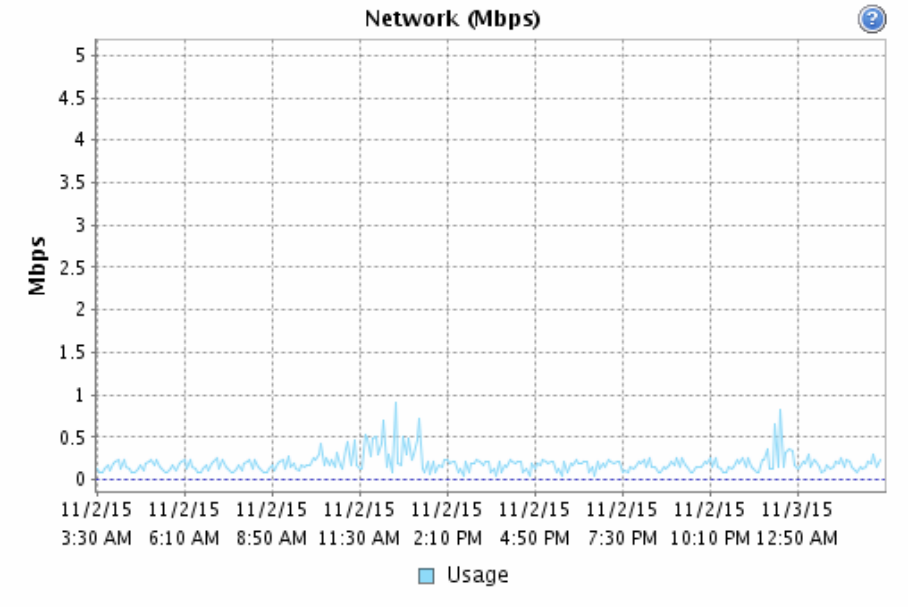

Figure 14: During the deployment of the virtual experiments no network operations are needed.

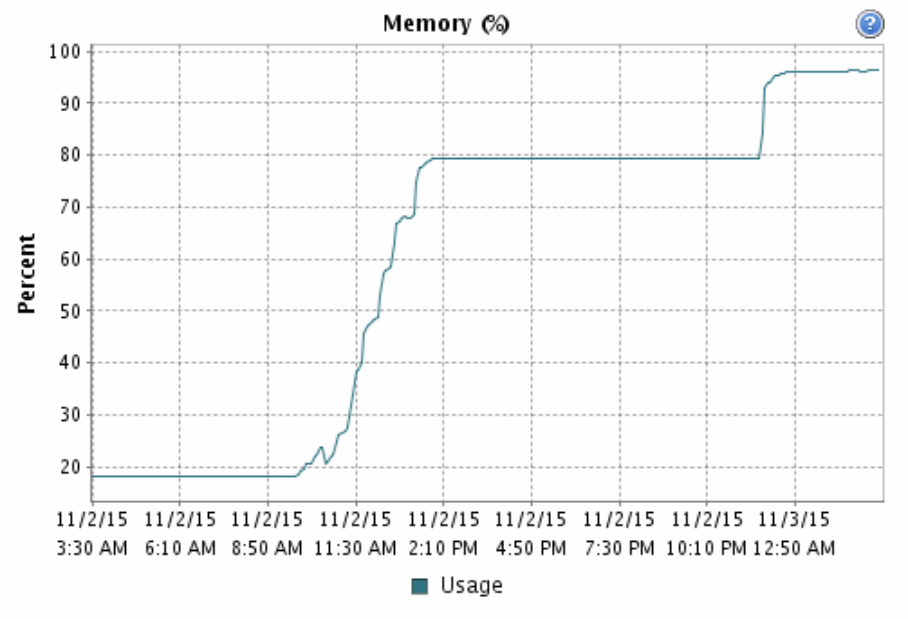

Figure 15: Allocated memory during the creation of the virtual experiments.

the idle times, the resource utilization is very low (below 10% CPU utilization). During active times, such as an experiment with 10 VMs (each of them allocates 1 core and 1GB RAM) actively applying DDoS attack to the servers using HPing3, we have observed that the CPU utilization has reached upto 100% (since there are two processors, the total percentage is 200%) as shown in Figure 16 for the 10 active VMs. And, similarly, there has been a peak where the memory utilization has been high. Since the attacker VMs were creating a large amount of network traffic, high amount of network operations have been seen but the disk operations were almost negligable during the attack time since no disk related operations were applied, as shown in Figure 17.

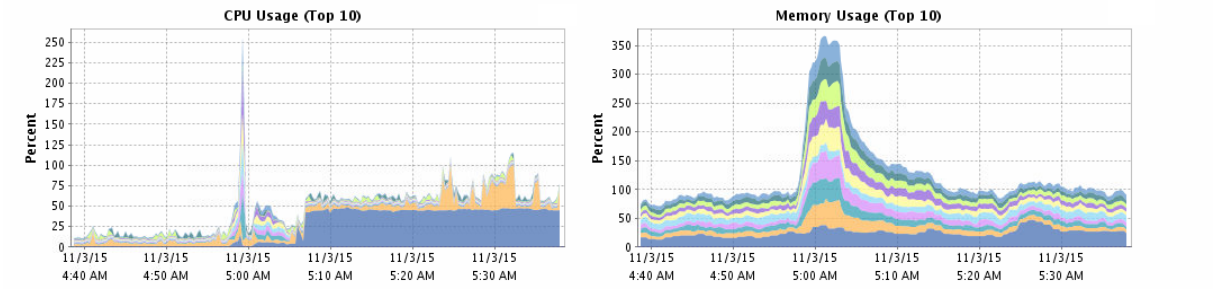

Figure 16: Network and disk utilization during an attack scnerio.

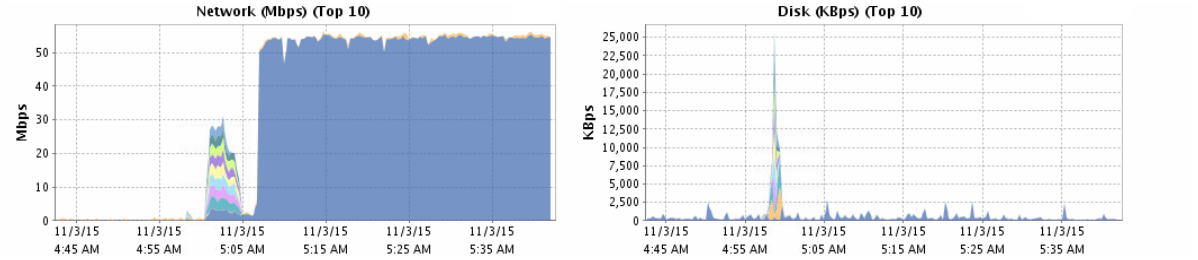

Figure 17: Network and disk utilization during an attack scnerio.

In addition to using CLaaS to provide all the cybersecurity experiments required for our cybersecurity class, we have developed a simplified version of these experiments for high school students in GenCyber camps that were held in summers of 2014 and 2015 at the UA Biosphere 2 [13]. The overall feedback from all the students that used CLaaS virtual cybersecurity experiments was very positive because of the easy to use Graphic User Interface (GUI) and the ability to perform the experiments from anywhere and anytime.

6 Conclusion and Future Work

Cybersecurity is a growing concern in today's IT world, especially due to the growing complexity, heterogeneous systems, and highly connected devices (computers, laptops, tablets, smart phones, smart watches, etc.) and lack of existing tools and algorithms to protect these resources against attacks and exploitations. Furthermore, currently we are facing a huge shortage in cybersecurity educational testbeds that can be used to demonstrate how vulnerabilities can be exploited, how attacks are launched, and how we can secure IT resources and services. In this work, we described the design and evaluation of Cyber-

security Lab as a Service (CLaaS). CLaaS enables students as well as researchers to run cybersecurity experiments such as DDoS attacks, password cracking, network packet sniffing, just to name a few. Our experimental evaluation of CLaaS and the feedbacks we received from students (college and high school students) all were positive. Current CLaaS implementation environment is limited by the resources we have – it can support upto 64 VMs per host machine to be used for the virtual cybersecurity labs. Therefore, in the future, we are planning to scale up the CLaaS by hosting it on public cloud services, such as AMAZON AWS, so that the cybersecurity lab services can be offered to a larger number of students from many schools across the country and world. Additionally using public cloud services will be supporting on demand and more flexible labs.

Acknowledgments

This work is partly supported by the Air Force Office of Scientific Research (AFOSR) Dynamic Data-Driven Application Systems (DDDAS) award number FA95550-12-1-0241, National Science Foundation research projects NSF IIP-0758579, NCS-0855087, and IIP-1127873, and Thomson Reuters in the framework of the Partner University Fund (PUF) project (PUF is a program of the French Embassy in the United States and the FACE Foundation and is supported by American donors and the French government).

References

[1] Vulnerabilities Type Distributions in CVE. Common Weakness Enumeration, Ver. 1.1, May 2007. `http://cwe.mitre.org/documents/vuln-trends/index.html`.

[2] Bind9. `http://www.bind9.net/`, 2015. [Online; Accessed on May 28, 2015].

[3] BOCHS. `http://bochs.sourceforge.net/`, 2015. [Online; Accessed on May 28, 2015].

[4] Cross site scripting. `http://www.acunetix.com/websitesecurity/cross-site-scripting/`, 2015. [Online; Accessed on May 28, 2015].

[5] dig – DNS lookup utility. `ftp://ftp.isc.org/isc/bind9/cur/9.10/doc/arm/man.dig.html`, 2015. [Online; Accessed on May 28, 2015].

[6] DNSSEC: DNS Security Extensions. `http://www.dnssec.net/`, 2015. [Online; Accessed on May 28, 2015].

[7] Linux bridge. `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`, 2015. [Online; Accessed on May 28, 2015].

[8] Linux KVM. `http://www.linux-kvm.org/`, 2015. [Online; Accessed on May 28, 2015].

[9] Nmap. `https://nmap.org/`, 2015. [Online; Accessed on November 1, 2015].

[10] Nmap scripts. `https://nmap.org/nsedoc/categories/dos.html`, 2015. [Online; Accessed on November 1, 2015].

[11] Qemu. `http://wiki.qemu.org/`, 2015. [Online; Accessed on May 28, 2015].

[12] Thc hydra. `http://sectools.org/tool/hydra/`, 2015. [Online; Accessed on May 28, 2015].

[13] The university of arizona, arizona gencyber cybersecurity camp. `http://gencyber.arizona.edu/`, 2015. [Online; Accessed on November 1, 2015].

[14] VMware. `http://www.vmware.com/`, 2015. [Online; Accessed on May 28, 2015].

[15] Wireshark. `https://www.wireshark.org/`, 2015. [Online; Accessed on May 28, 2015].

[16] Y. Al-Nashif, A. Kumar, S. Hariri, G. Qu, Y. Luo, and F. Szidarovsky. Multi-level intrusion detection system (ML-IDS). In *proc. of the International Conference on Autonomic Computing (ICAC'08), Chicago, Ilinois, USA*, pages 131–140. IEEE, June 2008.

[17] D. Atkins and R. Austein. Threat analysis of the domain name system (DNS). IETF RFC 3833, August 2004. http://www.ietf.org/rfc/rfc3833.txt.

[18] M. Blagov. DDoS Attacks. `https://www.incapsula.com/ddos/ddos-attacks/`, 2015. [Online; Accessed on November 1, 2015].

[19] H. Chen, Y. Al-Nashif, G. Qu, and S. Hariri. Self-configuration of network security. In *Proc. of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC'07), Annapolis, Maryland, USA*, pages 97–97, October 2007.

[20] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proc. of the 16th international conference on World Wide Web(WWW'07), Alberta,Canada*, pages 657–666. ACM, 2007.

[21] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *Proc. of the 2nd USENIX workshop on Hot topics in security(HotSec'07), Boston, USA*, volume 7, pages 10:1–10:6, August 2007.

[22] I. Habib. Virtualization with kvm. *Linux Journal*, 2008(166):8, 2008.

[23] T. Hornby. Sockstress. `https://defuse.ca/sockstress.htm`, 2015. [Online; Accessed on May 28, 2015].

[24] S. Kovach. We still dońt have assurance from apple that icloud is safe. `www.businessinsider.com/apple-statement-on-icloud-hack-2014-9`, 2015. [Online; Accessed on May 28, 2015].

[25] T. Matthews. Incapsula survey: What ddos attacks really cost businesses. `http://lp.incapsula.com/rs/incapsulainc/images/eBook%20-%20DDoS%20Impact%20Survey.pdf`, 2015. [Online; Accessed on May 28, 2015].

[26] Oracle Inc. Virtualbox. `http://www.virtualbox.org/`, 2015. [Online; Accessed on May 28, 2015].

[27] J. Sahoo, S. Mohapatra, and R. Lath. Virtualization: A survey on concepts, taxonomy and associated security issues. In *Proc. of the 2nd International Conference on Computer and Network Technology (ICCNT'10), Bangkok, Thiland*, pages 222–226. IEEE, April 2010.

[28] S. Sanfilippo. Hping 3. `http://www.hping.org/hping3.html`, 2015. [Online; Accessed on May 28, 2015].

[29] K. E. Stewart, J. W. Humphries, and T. R. Andel. Developing a virtualization platform for courses in networking, systems administration and cyber security education. In *Proc. of the Spring Simulation Multiconference (SpringSim'09), San Diego, Califonia, USA*, pages 65:1–65:7. Society for Computer Simulation International, March 2009.

[30] M. Tsugawa and J. A. Fortes. A virtual network (ViNe) architecture for grid computing. In *Proc. of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06), Rhodes Island, USA*, April 2006.

[31] C. Willems and C. Meinel. Tele-lab IT-security: an architecture for an online virtual IT security lab. *International Jornal of Online Engineering*, 4(2):31–37, 2008.

[32] C. Willems, O. Tringides, and C. Meinel. Practical IT security education with tele-lab. *The European Journal for the Informatics Professional*, 12(5):145–152, December 2011.

Author Biography

Cihan Tunc is a Research Assistant Professor and a member of Autonomic Computing Lab of the Electrical and Computer Engineering Department at the University of Arizona. He received his PhD from Electrical and Computer Engineering Department at the University of Arizona at 2015, his M.Sc. from Electrical and Computer Engineering Department at Northeastern University at 2010, and his B.Sc. from Electrical & Electronics Engineering Department at Bahcesehir University at 2008. His research interests include autonomic computing, power/energy and performance management of the cloud and large scale high performance computing systems, cyber-security, cyber-resiliency, big-data analytics, and Internet-of-Things (IoT).

Salim Hariri is a Professor in the Department of Electrical and Computer Engineering at The University of Arizona. He received his Ph.D. in Computer Engineering from University of Southern California in 1986, and an MSc from The Ohio State University in 1982. He is the UA site director of NSF Center for Cloud and Autonomic Computing and he is the Editor-In-Chief for the CLUSTER COMPUTING JOURNAL (Springer, http://clus.edmgr.com) that presents research techniques and results in the area of high speed networks, parallel and distributed computing, software tools, and network-centric applications. He is the Founder of the IEEE/ACM International Symposium on High Performance Distributed Computing (HPDC) and the co-founder of the IEEE/ACM International Conference on Cloud and Autonomic Computing. He is co-author/editor of four books on Autonomic computing, parallel and distributed computing: Autonomic Computing,: Concepts, Infrastructure, and Applications (CRC Press, 2007), Tools and Environments for Parallel and Distributed Computing (Wiley, 2004), Virtual Computing: Concept, Design and Evaluation (Kluwer, 2001), and Active Middleware Services (Kluwer, 2000). His research interests include autonomic cyber security, big data analytics, resilient cloud services, critical infrastructure protections, and autonomic programming, and resilient Dynamic Data Driven Application Systems (rDDDAS).