Cloud-based Labs and Programming Assignments in Networking and Cybersecurity Courses

Weiying Zhu
Department of Mathematical and Computer Sciences
Metropolitan State University of Denver
Denver, Colorado
wzhu1@msudenver.edu

***Abstract*—This is a full paper for innovate practice. Building a private cloud or using a public cloud is now feasible at many institutions. This paper presents the innovative design of cloud-based labs and programming assignments for a networking course and a cybersecurity course, and our experiences of innovatively using the private cloud at our institution to support these learning activities. It is shown by the instructor's observations and student survey data that our approach benefits learning and teaching. This approach makes it possible and secure to develop some learning activities that otherwise would not be allowed on physical servers. It enables the instructor to support students' desire of developing programs in their preferred programming languages. It allows students to debug and test their programs on the same platform to be used by the instructor for testing and grading. The instructor does not need to spend extra time administrating the computing environments. A majority (88% or more) of the students agree that working on those learning activities in the private cloud not only helps them achieve the course learning objectives, but also prepares them for their future careers.**

Keywords—Private Cloud; Public Cloud; Hands-On Laboratory; Network Programming; Security Programming

I. INTRODUCTION

Hands-on labs and programming assignments are very helpful for students to better achieve the learning objectives in various computer science (CS) courses. In some CS courses, there is a need to write programs that run on one or multiple powerful servers. The skills of remotely connecting to and working on Unix/Linux servers are critical for the software development of various Internet services. Moreover, the needs in some networking or cybersecurity learning activities may threaten the security of the networks and physical servers on campus. Recently, educators have been exploring the use of a public cloud to address these needs. Probably because of the AWS in Education Teaching Grants, which have been retired in favor of AWS Educate , the AWS cloud has been used in several CS courses such as parallel computing [6], distributed programming [4], database [3], and security [9, 12]. A public cloud in CS courses is a promising solution because of its accessibility, scalability, flexibility, and security. We once used a public cloud for teaching [14] and also appreciate its benefits to teaching and learning. Public cloud services have also been utilized to address various educational needs in non-CS courses. Examples include the use of Google Forms and Google Sheets in a mechanical engineering course [13], the use of the Elastic Beanstalk in the AWS cloud as the backend of a virtual reality pedagogical ecosystem [1], and the use of Cloud Bees or Openshift in communications and control system courses [5].

Due to the fast growth of the computing power and data storage, the decreased cost of computing and storage devices, the advances in cloud computing, and the development of a wide spectrum of Computational Environments (CEs) [10], building a private cloud has been explored and proved to be feasible and cost-effective in higher education institutions [2, 7, 10, 11]. Riliskis *et al*. discussed the technical readiness and challenges of using cloud technology for university scale IT-infrastructure in [7]. Rivera *et al*. proposed an approach that integrates cloud-based smart adaptive remote laboratories with virtual learning environments in a digital electronics course [8]. The benefits of hosting virtualized CEs in a private or public cloud are assessed in [10]. Eckroth presented their experiences of building and using a virtual cluster (typically a key part of most private cloud computing architectures) in a big data mining and analytics course at a small private liberal arts college in [2]. Shoop *et al.* described the effectiveness of using virtual clusters in small colleges to teach parallel and distributed computing in [11]. The cost analysis in [2] shows that a physical cluster is much more expensive than a virtual cluster or the service in the AWS cloud. The analysis in [10] shows that the cost of the virtual instances in the AWS cloud will exceed the cost of the on-premises machines for hosting equivalent virtualized CEs in the private cloud in 17 months.

In September 2015, the Information Technology Services (ITS) at our institution started providing Virtual Computing Environments (VCEs) in the private cloud, which are virtual servers, virtual networks, virtual labs, or virtual classrooms that are customized per the instructor's request. We have developed cloud-based hands-on labs and programming assignments to engage students in learning and help them achieve the learning objectives in two CS courses, networking and cybersecurity. We found that the use of VCEs in the private cloud enables students to 1) use multiple virtual servers with different IP addresses for debugging and testing client and server programs, 2) use various transport-layer ports, which would otherwise be blocked if physical servers were used, in their programs, 3) remotely connect to, work on, and transfer files to Linux virtual servers at anytime from anywhere, 4) work on security hands-on activities that would otherwise not be allowed due to the potential threat to a physical server, 5) debug their programs on the Linux platform on which the instructor tests programs, such that they may find and fix the platform/environment associated

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

bugs in their programs that may work in the Integrated Development Environments on their Windows computers, and 6) develop programs in their preferred programming languages.

The main contributions of this paper are our novel design of cloud-based hands-on labs and programming assignments in networking and cybersecurity courses, and our experience of using the virtual servers in the private cloud at our institution to support those learning activities since Spring 2016. This paper also introduces the major hardware for supporting VCEs in the private cloud and the configuration of the virtual servers for our courses. A student survey was given in class at the end of every semester to get students' opinions on the use of the private cloud for their hands-on labs and programming assignments as well as the impact to their learning. To evaluate our approach, the instructor's observations and the analysis of the survey data are discussed. This approach enables the instructor to better meet students' needs in learning. It also allows instructors to focus on developing and evaluating the cloud-based learning activities while the ITS administrates the VCEs. It is shown by the survey data that our approach has a strong positive impact to student learning experiences. Most (88% or more) of the students are in favor of using virtual servers not only in these two course but also in other CS courses whenever applicable, and agree that such experience is helpful for their future career developments.

Although the use, access, administration, and cost of a public cloud are not the same as that of a private cloud, our hands-on labs and programming assignments can be adapted and adopted in a public cloud such as the AWS cloud. AWS EC2 (Elastic Cloud Computing) instances can be customized and used as the virtual servers to support those learning activities.

The remainder of this paper is organized as follows. Section 2 discusses the relevant resources in the private cloud at our institution that have been used to support the cloud-based labs and programming assignments designed for our networking course and cybersecurity course, and how students access and work on the virtual servers set up for these two courses. Section 3 presents the student learning objectives that are directly supported by the cloud-based learning activities in these two courses, and the design of those cloud-based hands-on labs and programming assignments. Both the instructor's observations and student survey data are analyzed in Section 4. Finally, Section 5 summarizes and concludes this paper.

II. The Private Cloud

The ITS creates and configures the virtual servers, virtual networks, virtual labs, or virtual classrooms in the private cloud according to the instructors' various academic needs. The VCEs are deleted after the semester is over.

A. Relevant Resources in the Private Cloud

In the data center on campus, ITS uses two Hewlett Packard Proliant BL 460c Gen9 servers, each of which has two 12-core processors and a memory of 256GB, 4 TB from the storage area network, and networking devices to support the custom VCEs.

We requested virtual servers for our networking course in Spring 2016 and Spring 2017 and for our cybersecurity course in Fall 2016 and Fall 2017. The operating system on the virtual servers is Ubuntu 16.04. OpenJDK Runtime Environment, Python, GCC (the GNU Compiler Collection), and .NET Core SDK were installed to support programming in Java, Python, C++, and C#. ITS created an user account for every student on each virtual server, who has read/write/execute permissions only in his or her home directory. ITS also created an account for the instructor. The read/write/execute permissions were given to the instructor not only in the instructor's home directory but also in all students' home directories. The sudo access was granted to the instructor for the "*passwd*" command, which allows the instructor to reset students' passwords, the "*kill*" command, which allows the instructor to kill the processes left suspended by students to release system resources such as transport-layer ports, and all the "*apt*" commands, which allows the instructor to install additional software packages if needed.

Due to the firewall configuration inside our campus network, the virtual private network (VPN) connection is required to allow the access to virtual servers from outside the campus network. While using off-campus networks for connecting their computers to the Internet, students must first connect their computers to the VPN before accessing virtual servers. Per the instructor's request, at the beginning of a semester, ITS creates a student VPN account for every student enrolled in the course.

B. Accessing and Working on Virtual Servers

On a Mac computer, we use the "*ssh*" command, which starts an OpenSSH SSH (Secure Shell) client program, to log into a remote virtual server and work on this virtual server. Once logging into a remote virtual server, students run command-line commands to navigate the file system in their home directories, edit files, compile the source code files of their programs, and debug and test their programs. Fig. 1 shows the connection to a remote virtual server from a Terminal on a Mac computer, and the execution of command-line commands on this virtual server.

Fig. 1. Connecting to and working on a virtual server via ***ssh***.

The "*sftp*" command is used to establish a secure file transfer connection to a remote virtual server from a local Mac computer, upload files to this virtual server, and download files from this virtual server. "*ssh*" and "*sftp*" commands are also available on

Linux computers. On a Windows computer, PuTTY, a free SSH client for Windows, is used to connect to and work on a remote virtual server. PSFTP is used for secure file transfer between a local Windows computer and a remote virtual server.

Fig. 2 shows a typical setup for students to work on one or multiple virtual servers in the private cloud. Students may remotely work on a single virtual server for hands-on labs or programming. For socket programming assignments and SSL (Secure Socket Layer) programming assignments, students may run their server programs on one virtual server and run their client programs on the second virtual server and their computers. This setup allows students to run server and client programs on different computers in a real networked environment. In the cryptography project, a student may execute the sender's program on one virtual server, and then transfer its output file(s) to the other virtual server. A teammate may run the receiver's program on the other virtual server to decrypt the cypher text, verify the hash value, and recover the message file.

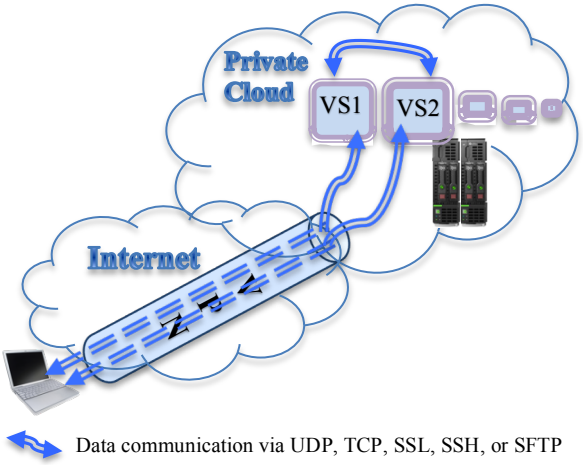

Fig. 2. Using multiple virtual servers (VS) for labs and programming.

III. Course Design

In the networking course, students work on the virtual servers to learn application-layer protocols, common network applications, socket programming, and routing algorithms. In the cybersecurity course, students use the virtual servers to study (1) the use of Java Cryptography Architecture (JCA) for symmetric encryption, message authentication, public-key encryption, digital signature, and digital envelope, (2) the use of Java Secure Socket Extension (JSEE) for SSL programming, and (3) buffer overflow vulnerabilities in C programs.

A. Relevant Student Learning Objectives

The cloud-based learning activities described in the remainder of this section were designed to directly support all of the learning objectives listed below. The student learning objectives that are not directly supported by those cloud-based learning activities are not included here.

In the networking course, the *most relevant* objectives are

O1. Understand various application-layer protocols,

O2. Use common Internet applications such as *ssh* and *sftp*,

O3. Compare and contrast connection-oriented and non-connection-oriented services,

O4. Compare and contrast algorithms for packet routing,

O5. Create Internet applications using net-centric programming techniques.

In the cybersecurity course, the *most relevant* objectives are

O1. Describe the methods for defending against attacks,

O2. Explain how cryptography works and how to apply it to achieve confidentiality and integrity,

O3. Develop security programs using SSL sockets,

O4. Evaluate fixes for system and network attacks.

B. Hands-on Labs in the Private Cloud

Several hands-on *labs* have been designed in these two courses to 1) help students learn how to access and work on virtual servers, and transfer files between a local computer and a virtual server, and 2) utilize the resources in our private cloud to help students achieve the relevant learning objectives.

The first lab, which is given in both courses at the beginning of a semester, is designed to introduce the resources in the private cloud and the tools of accessing and using the VCEs created for our course. Task 1 is for every student to connect to the VPN using a personal account. Task 2 is to securely connect to virtual servers via "***ssh***" and "***sftp***" commands on a local Mac computer or via PuTTY and PSFTP clients on a local Windows computer. Task 2 also includes practicing common command-line commands for file operation and program execution in secure shell and secure file transfer windows. Task 3 is to download the networking or cryptographic programs provided by the instructor and test them on virtual servers. Each student is assigned with a unique port number to run his or her server program on a virtual server that is shared by the whole class. This lab prepares students in both courses for accessing and using the resources in the private cloud and helps students achieve learning objective *O2* listed for the networking course.

In our cybersecurity course, the second lab is designed for the students to study buffer overflow cases on virtual Linux servers. The instructor provides C programs (e.g., `bufferX.c` shown in Fig. 3) that contain buffer overflow vulnerabilities, which are caused by either the use of unsafe C library functions or careless programming practices. Students study these C programs to exploit the vulnerabilities, design testing cases to cause buffer overflow, and execute those programs on virtual servers to cause "core dumped" or "segment fault" errors. Next, students are required to remove buffer overflow vulnerabilities in those C programs by replacing the unsafe C library functions with their safer alternatives or by correcting programming errors. Fig. 3 (a) and (b) demonstrate two buffer overflow cases of the same C program `bufferX.c` that are caused by (a) overflowing the char array `buf[]` that is declared inside the function `main()`, and (b) overflowing the char array `info[]` that is declared in the function `sentence()`. Fig. 3(c) shows the execution of `bufferX-fix`, which is a corrected version

of `bufferX` by fixing all buffer overflow vulnerabilities. This lab was designed to help students achieve the learning objectives *O1* and *O4* listed above for the cybersecurity course.

In the third lab of our cybersecurity course, students learn how to create JKS (Java KeyStore) entries and use them in their Java programs that use JSEE. Students first connect to the virtual servers and use the "***keytool***" command to create a *key entry* in a JKS key store and its corresponding *trusted certificate entry* in a JKS trust store. Next, students test example SSL Server and Client programs, which require server authentication, on the virtual servers. The JKS key store is used by the server program and the JKS trust store is used by the client program. This lab helps students in the cybersecurity course achieve learning objectives *O1*, *O2*, and *O3* that are listed above.

```
[[redacted]1@cs[redacted]:~/lab2$ ./bufferX
Please input your address:
[99 Maple Street
In sentence(), sizeof info[] is 36, strlen of info[]
is 33, info[] is
Hello!  Muffin Man      99 Maple Street
bufferX sentence() done.
In main(), sizeof buf[] is 50, strlen of buf[] is 59,
 buf[] is
Hello! My name is Muffin Man and I live at 99 Maple S
treet.
bufferX main() done.
*** stack smashing detected ***: ./bufferX terminated
Aborted (core dumped)
[[redacted]1@cs[redacted]:~/lab2$
```

(a) Overflow of the array `buf[]` declared in the `main()` function.

```
[[redacted]1@cs[redacted]:~/lab2$ ./bufferX
Please input your address:
[12345 Magic Kingdom Ave
In sentence(), sizeof info[] is 36, strlen of info[]
is 41, info[] is
Hello!  Muffin Man      12345 Magic Kingdom Ave
bufferX sentence() done.
*** stack smashing detected ***: ./bufferX terminated
Aborted (core dumped)
[[redacted]1@cs[redacted]:~/lab2$
```

(b) Overflow of the array `info[]` declared in the `sentence()` function.

```
[[redacted]@cs[redacted]:~/lab2$ ./bufferX-fix
Please input your address:
[12345 Magic Kingdom Ave
In sentence(), sizeof info[] is 36, strlen of info[]
is 35, info[] is
Hello!  Muffin Man      12345 Magic Kingd
bufferX sentence() done.
In main(), sizeof buf[] is 50, strlen of buf[] is 49,
 buf[] is
Hello! My name is Muffin Man and I live at 12345
bufferX main() done.
[[redacted]@cs[redacted]:~/lab2$
```

(c) Fixing the buffer overflow vulnerabilities in `bufferX.c`.

Fig. 3. Demo of Lab 2 "Buffer Overflow Vulnerabilities" on a virtual server.

C. Programming on a Single Virtual Server

Two network programming assignments (one for the Dijkstra's algorithm and the other for the Distance Vector algorithm) and one security programming assignment (RBAC: Role Based Access Control) have been designed for students to debug and test their programs in a command-line environment on a single virtual Linux server in the private cloud.

In the assignment for the Dijkstra's algorithm, students are required to implement the logic of discovering the shortest path tree from a source router to all the other routers in the network, and the logic of generating the forwarding table that includes one outgoing link for each destination router. The execution of a student's program written in Python for this assignment is shown in Fig. 4. In this demo, V0 through V5 are the routers in the network and V0 is the source router. In the assignment for Distance Vector algorithm, students are required to implement how a source router recalculates its link vector and distance vector in response to an event of the change of the link cost to a neighboring router or an event of receiving an updated distance vector from a neighboring router. These two assignments were designed for the learning objective *O4* in the networking course.

```
[[redacted]@cs[redacted]:~/StuWork_HW7$ python3 dijkstra.py
[What is the number of Routers: 6
Reading links & their costs from file <topo.txt>...
.success.
Cost Matrix:
     0    1    2    3    4    5
0    0    5    1    3    3    -
1    5    0    2    -    -    -
2    1    2    0    1    -    -
3    3    -    1    0    2    1
4    3    -    -    2    0    2
5    -    -    -    1    2    0
After round 5:
Y':     [  (0, 2)  (2, 3)  (2, 1)  (0, 4)  (3, 5) ]
N':     [  0   2   3   1   4   5  ]
D(i):   [  -   3   1   2   3   3  ]
p(i):   [  -   2   0   2   0   3  ]
Forwarding table from node V0
Destination:      Link:
 V1                (V0, V2)
 V2                (V0, V2)
 V3                (V0, V2)
 V4                (V0, V4)
 V5                (V0, V2)
[[redacted]@cs[redacted]:~/StuWork_HW7$
```

Fig. 4. Demo of a student's program in Python for the "Dijkstra's Algorithm" programming assignment on a virtual server.

In the assignment for RBAC, students are required to implement a simplified NIST RBAC model, which includes limited role hierarchy, inheritance of permissions, static separation of duty relations, user-role matrix, and a few administrative functions. This programming assignment aims at the learning objective *O1* in the cybersecurity course.

For every programming assignment, students are required to upload their programs to a virtual server via secure file transfer, connect to the virtual server via a secure shell client, and compile and test their programs on the virtual server. Therefore, these programs use command-line I/O. Because of the installation of OpenJDK, Python, GCC, and .NET Core SDK on the virtual servers, students may freely choose to develop their programs in Java, Python, C, C++, or C#.

D. Programming on Multiple Virtual Servers

In some programming assignments, it is beneficial to learning for students to use multiple virtual servers for debugging and testing their programs. Three *socket*

programming assignments, one *cryptography project*, and one *SSL programming assignment* have been designed for students to utilize multiple virtual servers in the private cloud.

In three socket programming assignments in the networking course, students are required to develop a pair of Query Client and Server programs, a pair of simplified HTTP (Hypertext Transfer Protocol) Client and Server programs, and a pair of simplified SMTP (Simple Mail Transfer Protocol) Client and Server programs, respectively. In their Query programs, students are required to use UDP sockets for the Client program to send an item ID to the Server program and the Server program to respond with the detailed information. In their HTTP or SMTP programs, students are required to use TCP sockets in the Client and Server programs. The HTTP Client program constructs an HTTP request message using the information collected from user inputs, sends this message to the HTTP Server program, receives and displays the HTTP response message, and saves the body part enclosed in the HTTP response message into a text file if any. The HTTP Server program constructs an HTTP response message upon receiving the HTTP request message, encloses the requested text file if the request method is "GET" and the text file exists, and sends this response message back to the HTTP Client program. The SMTP Client program collects all the information regarding an email via user inputs, constructs and sends SMTP command messages and the Mail message to the SMTP Server program, one at a time after receiving the response message to the previous command/Mail message, and displays each response message. Following the 3-phase data-transfer procedure, the SMTP Server program responds each SMTP command/Mail message. If the incoming SMTP command message is invalid or out of order, the SMTP Server program responds with a "503 5.5.2" response message to ask for the correct command message. The SMTP Server displays all the command messages and the Mail message. Fig. 5 demonstrates the execution of a student's SMTP client and server programs on two virtual servers for establishing the TCP connection and sending one email from the client to the server.

```
[redacted]:/home/[redacted]/HW04/Client$ java HW4_SMTPClient
Please enter the DNS or IP where the SMTP Server Program runs:
[redacted].edu
220 [redacted].87
Please enter sender's email address: alice@abc.edu
Please enter receiver's email address: bob@xyz.com
Please enter the subject of the email: 1st email for testing
Please enter the email body that ends with <CRLF>.<CRLF>:
Once upon a time, ......
The end.
.
250 [redacted].87 Hello [redacted].69
250 2.1.0 Sender OK
250 2.1.5 Recipient OK
354 Start mail input; end with <CRLF>.<CRLF>
250 Message received and to be delivered
```

```
[redacted]:/home/[redacted]/HW04/Server$ java HW4_SMTPMultiServer
HELO abc.edu
MAIL FROM: alice@abc.edu
RCPT TO: bob@xyz.com
DATA
To: bob@xyz.com
From: alice@abc.edu
Subject: 1st email for testing

Once upon a time, ......
The end.
```

Fig. 5. Demo of a student's SMTP client and server programs in the cloud.

These three programming assignments are specifically designed for the learning objectives *O1*, *O3*, and *O5* listed above for the networking course.

In the cryptography project designed for the cybersecurity course, there are three options for students to choose from. In all three options, students are required to develop a key generation program, a sender's program, and a receiver's program to pursue integrity and confidentiality when the sender sends a plaintext message (M) read from a local file to the receiver. The key generation program first randomly generates two pairs of RSA public and private keys (one pair for sender and the other for receiver) and save the modulus and exponents of each public or private key into a key file. This program then takes a 16-character user input and save its UTF-8 code as the 128-bit AES symmetric key into a key file.

In Option 1, the sender's program calculates the *authentic message digest* of M using SHA256 for hashing and AES for encryption, and then encrypts M together with its *authentic message digest* via RSA. The receiver's program decrypts the received cypher text using the receiver's RSA private key, calculates the SHA256 hash value of M, and compares it with the hash value that is decrypted from the received *authentic digital digest* of M using the AES symmetric key. In Option 2, the sender's program calculates the *digital signature* of M using SHA256 for hashing and RSA for encryption. Then the sender's program uses AES to encrypt M together with its *digital signature*. The receiver's program decrypts the received AES cypher text, calculates the SHA256 hash value of M, and compares it with the hash value that is decrypted from the *digital signature* of M using the sender's RSA public key. Option 3 integrates the design of *keyed-hash message authentication code (HMAC)* and the design of *digital envelope*. The sender's program calculates the HMAC of M via SHA256 using the AES symmetric key, calculates the AES cypher text of M, and calculates the RSA cypher text of the AES symmetric key using the receiver's RSA public key. The receiver's program decrypts the RSA cypher text using the receiver's RSA private key to get the AES symmetric key, decrypts the AES cypher text to get M using the AES symmetric key, calculates the HMAC of M, and compares it with the HMAC of M received from the sender.

Fig. 6 shows a receiver's program developed by a team for this cryptography project (Option 2). "message.aescipher" is the AES cipher text of M and its digital signature generated by the sender's program. After being copied to the receiver's side, one byte in this file was modified. Consequently, the locally calculated hash is not the same as the received hash, and some bytes in the resulting message M in "Garden.out.txt" are different from that in the original message M in "Garden.txt".

No matter which one of the three options they choose for the sender's and receiver's programs, students will learn the APIs in the language of their choice for SHA256 hashing, AES symmetric encryption and decryption, and RSA public key encryption and decryption. Since there is no limitation on the size of the local file from which the sender's program reads M, for AES encryption and RSA encryption, the sender's program is expected to select a block cipher such as CBC (Cipher Block Chaining) or a stream cipher such as CFB (Cipher Feedback) as the mode of operation, read multiple plaintext blocks of M (one

at a time), calculate the cipher text of each plaintext block, and keep writing those cipher text blocks to the cipher text file. This cryptography project is designed to support the learning objectives *O1* and *O2* in the cybersecurity course.

```
[redacted]:Receiver [redacted]$ diff message.aescipher ../Sender/message
.aescipher
Binary files message.aescipher and ../Sender/message.aescipher di
ffer
[redacted]:Receiver [redacted]$ java Receiver
Read from symmetric.key: key=0123456789ABCDEF
Read from XPublic.key: modulus = 136563429211385522234093701178849661598155862553939492989058063998535835186001565233622271353795280026297885861864753997929405535870090579756414770772686577591286206612322098308800948352511116095303441684074357308528374129271035517767038727943580787547154066033172495192383570194003470061065752814794305701567, exponent = 65537
Input the name of the output message file: Garden.out.txt
Cipher Text of Digital Signature:
BD 93 9D 76 13 72 73 8E 2F F3 92 49 7C FF B2 26
FB B9 DD DA 47 97 F5  6  2 9F AF C3 E6 5A 65 D5
AB 9E 70 AF 8C 27 F9 21 C4 3E 57 C7 AA 67 32 BB
 4 BF 19 57 24 D5 42 83 25 D8 DC 3E 40 C5 46 17
EE  B 67 D8 34 D4  C 55 3A 7F B1 64 83 30 B9 50
E4 90 8E 6B F1 16 81 A6 C0 A2 AC 36 D0 79 7C 3E
90 FE 80 18 BA  6 47  E 59 70 31 A7  E D1 BC  8
7E B6 FF 59 99 91 13 6F 3B A3 8F 7F 89 D0 AA BF
Received hash:
83 58 7F D3 C6 C4 DB 51 78 B1 1C 60 76 FC 85 AB
C2 1A 7F 65 F5 9D 63 48 46 E1 D2 83 47 1B 47 E4
Locally calculated hash:
33 DE EF 59 EF A1 59  1 EC DF BB 9F DA E6  A  A
8A FE DA B3 62 16 BA 60 73 9A AA C6 8E  5 B1  6
Altered
[redacted]:Receiver [redacted]$ diff Garden.out.txt ../Sender/Garden.txt
10c10
< suggested that!'o??w{?x?M??apital place for a garden".
---
> suggested that it would be a "capital place for a garden".
86d85
< g
\ No newline at end of file
```

Fig. 6. Demo of a team's receiver's program for the cryptography project.

In the SLL programming assignment in the cybersecurity course, students are required to write a Server program and a Client program, which establish a SSL connection with each other and use SSL sockets to send messages to each other. Students need to use the "***keytool***" command to create a *key entry* in a JKS key store and its corresponding *trusted certificate entry* in a JKS trust store. The Server program uses the JKS key store and the Client program uses the JKS trust store, respectively, for server authentication, confidentiality, and integrity. This assignment aims at the learning objectives *O1* and *O3* listed above in the cybersecurity course.

IV. Assessment

The instructor's observations and student surveys are used to evaluate the use of the private cloud to support student learning in our networking course and cybersecurity course.

A. The Instructor's Observations

The instructor once used virtual (EC2) instances in the AWS cloud in the networking course in Spring 2014 [14], used a departmental physical server in the cybersecurity course in Fall 2015, and has been using virtual servers in our private cloud since Spring 2016. Based on the instructor's observations, Table 1 records the properties of a private cloud, a public cloud, and physical servers while they are used for hands-on labs and programming assignments in networking and cybersecurity courses. Although our intention is not to argue the advantages and disadvantages of those three approaches, we observe that a private or public cloud can better meet the needs in learning.

A physical server is a powerful computer that is shared by multiple instructors and/or groups in the department for various purposes. The confidentiality, integrity, and availability of the data on a physical server need to be well protected. The failure of a physical server caused by malicious may have a big impact. Due to high security concerns, a departmental physical server resides in a well-protected subnet, in which the vast majority of the transport-layer ports, except for a few ports used by system programs such as SSH and SFTP, are blocked by the firewall. Thus, a student's client and server programs running on different computers cannot send messages to each other via any transport-layer port. This prevents physical servers from supporting some or all of the tasks in the first lab in both courses, the third cybersecurity lab, three socket programming assignments in the networking course, and one SSL programming assignment in the cybersecurity course. On the contrary, there are much less security concerns on virtual servers in a private cloud and virtual instances in a public cloud. The virtual servers/instances in a cloud are used only by the students in one class, they do not host any critical service, they can be deleted and re-created with little cost whenever anything goes wrong, and they are re-imaged or re-created before a semester starts. Therefore, those virtual servers/instances reside in subnets in which the transport-layer ports are left wide open by the firewall. This allows students to use various ports for socket programming.

Virtual servers in a private cloud or physical servers normally have high CPU rate and large memory and, therefore, are shared by all the students in a class, each having access only to his or her home directory. They are kept running 24/7 throughout the semester. For grading, the instructor just need to connect to one or two virtual or physical servers and navigate into students' home directories to test their programs. For cost saving and easy management, virtual instances in a public cloud are configured with low CPU rate and small memory. They are created for and used by each student individually. Students are required to stop their individually assigned virtual instances in a public cloud after each use to save cost. To test and grade students' programs, the instructor starts and connects to all virtual instances (two per student) and stop them afterwards.

VPN account and connection are required for accessing virtual servers in the private cloud or departmental physical servers from outside the campus network. Virtual instances in a public cloud can be accessed anytime from anywhere without a VPN connection. From the instructor's perspective, there was a nontrivial administration overhead while using the virtual instances in a public cloud. The instructor used to spend extra time for creating student user accounts, configuring and creating virtual instances (two for each student), and monitoring whether virtual instances were shut down in time to avoid unnecessary charge. Now, the ITS takes care of all the administration work.

No matter whether virtual servers in a private cloud, virtual instances in a public cloud, or physical servers are used, students debug and test their programs on the same platforms that are used by the instructor for testing and grading. This solves the problems caused by inconsistent programming environments or

TABLE I. PROPERTIES OF A PRIVATE CLOUD, A PUBLIC CLOUD, AND PHYSICAL SERVERS WHILE BEING USED FOR LEARNING ACTIVITIES

Property	Private Cloud	Public Cloud	Physical Servers
Security concerns	Low	Low	High
Re-image virtual/physical servers/instances	Every semester or whenever necessary	Every semester or whenever necessary	Normally not re-imaged once they are set up
Transport-layer ports	Widely open	Widely open	Blocked except a few ports used by system programs
Support to labs/homework	Full support to all activities	Full support to all activities	Limited, no support to some
CPU rate per server/instance	High	Low	High
Memory per server/instance	Large	Small	Large
Share servers/instances	Yes, *shared* by students in *one single class*.	No, *each individual* student is assigned two virtual instances.	Yes, *shared* by students in *multiple courses*.
Keep servers/instances running 24/7	Yes	No, stop virtual instances after each use to save cost.	Yes
Require VPN connection	Yes, for access from outside the campus network.	Not at all	Yes, for access from outside the campus network.
Number of servers/instances the instructor connects to	Two for the whole class	Many (two per student)	One for the whole class
Who creates/administrates servers/instances/accounts	University ITS	The instructor	Department technician
Cost model	High upfront cost for hardware. The cost per virtual server is much lower than that of a physical server [2].	Very low cost per use for each virtual instance. The cost over time will exceed that of the on-premises machines in the private cloud (e.g., in 17 months [10]).	High upfront cost for hardware.

versions. It allows the instructor to support students' desire of writing programs in their preferred programming languages. So far, students have chosen to program in Java, Python, C, C++, and C# in these two courses.

B. Student Survey Results

Both the networking course and the cybersecurity course are upper-division elective courses in our CS curriculum, and have been mostly taken by junior and senior undergraduate students for their CS majors. They have also been taken by a very small percentage of students for their CS minors or Individualized Degree Programs. In the near future, we expect to see a few students taking these two courses for the newly approved Computer Engineering major as electives. A student survey was given in class at the end of every semester. 37 students (18 out of 22 in Spring 2016 and 19 out of 25 in Spring 2017) returned their answers in the networking course. 24 students (13 out of 20 in Fall 2016 and 11 out of 19 in Fall 2017) returned their answers in the cybersecurity course.

There are ten questions to survey students' experiences of using virtual servers in our private cloud for hands-on labs, programming assignments, and projects, and the impact to their learning. Below is a concise version of the questions.

- ***Q1***: It is easy to learn how to use the virtual servers and the VPN to accomplish the relevant tasks.
- ***Q2***: Using the virtual servers to perform the assigned tasks is not more difficult than using my computer or a lab computer.
- ***Q3***: Using the virtual servers allows me to set up the running environment of my programs and access it anywhere.
- ***Q4***: Using the virtual servers allows me to test my client/server programs in a real networked environment.
- ***Q5***: Using the virtual servers in this course is beneficial for me to accomplish course learning objectives.
- ***Q6***: Using the virtual servers is helpful for improving my skills on cloud computing.

- ***Q7***: Using the virtual servers is helpful for improving my skills of programming and running programs on Linux/Unix.
- ***Q8***: Overall, I am favor of using virtual servers in this course.
- ***Q9***: I would like to use virtual servers or other cloud services in other CS courses whenever appropriate.
- ***Q10***: The experience of using virtual servers in this course is helpful to my career development.

The survey results are illustrated in Fig. 7 for the networking course and in Fig. 8 for the cybersecurity course. First of all, we can see that although more than 80% of the students in both courses strongly or somewhat agree that it is easy to learn how to use virtual servers and the VPN (Q1), the percentage of students who agree that using virtual servers is not more difficult than using their computers or lab computers (Q2) is only slightly higher than 60% in both courses. Also, only 20% of the students in the cybersecurity course strongly agree with Q1. This indicates that there is some extra work or difficulty for students to upload, debug, and test their programs on virtual servers.

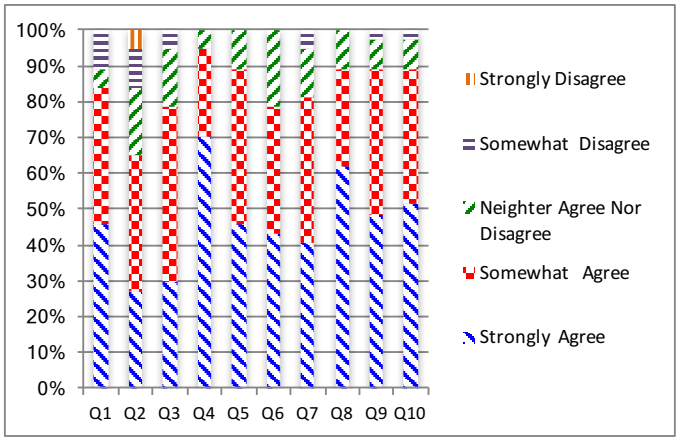

Fig. 7. Results of Q1 through Q10 in the networking course.

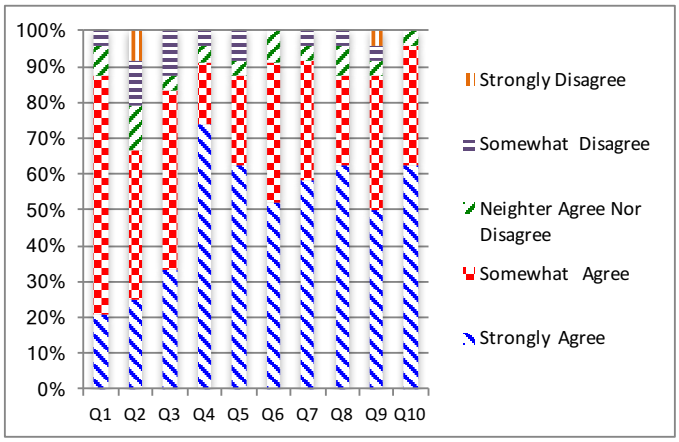

Fig. 8. Results of Q1 through Q10 in the cybersecurity course.

78% of the students in the networking course and 83% of the students in the cybersecurity course agree with the accessibility to the services in our private cloud (Q3). Connecting to the VPN and connecting to the virtual servers could be confusing. On campus, students can connect to the virtual servers from their laptops without connecting to the VPN. Every semester, there were always some students indicating in the survey or telling me in person that they could never connect to the virtual servers at home although they could do it on campus. It turned out that they didn't first connect their laptops to the VPN before trying to connect to the virtual servers at home because they thought the VPN and virtual servers were the same thing. Therefore, more efforts need to be made in addition to the first lab to clarify that VPN and virtual servers are different services in a private cloud. For example, in the future, the instructor will add such information in every programming assignment or project.

Next, we can see that 88% or more of the students in both courses strongly or somewhat agree that using virtual servers allows them to test their programs in the real networked environment (Q4) and is beneficial for them to accomplish the learning objectives (Q5), they are favor of using virtual servers in this course (Q8), they would like to use the services in the private cloud in other CS courses whenever appropriate (Q9), and this experience is helpful for their future careers (Q10).

Moreover, about 80% of the students in the networking course and slightly more than 90% of the students in the security course strongly or somewhat agree that using the virtual servers is helpful for improving their knowledge of cloud computing (Q6) and their programming skills on Linux/Unix (Q7). A possible reason is that in the cybersecurity course, besides programming, we also use the virtual servers for other activities such as data encryption and decryption, message authentication, JKS key generation via both the "***keytool***" command and Java APIs, and the case study of buffer overflow vulnerabilities.

V. Conclusions

In summary, we developed cloud-based hands-on labs and programming assignments/projects in two upper-division CS courses: networking and cybersecurity. We have been using the services in the private cloud at our university as the platform to support those learning activities. Both the instructor's observations and student survey data demonstrate the positive impact of this approach to learning and teaching. The instructor is in favor of this approach because of its positive impact to student learning, security, accessibility, convenience, and little administration overhead. Most students also agree that the use of the private cloud for those hands-on labs and programming assignments/projects is beneficial to their learning and future careers, and they are in favor of using virtual servers not only in this course but also in other CS courses whenever appropriate. Although the use, access, administration, and cost of a public cloud are normally different from that of a private cloud, the cloud-based hands-on labs, programming assignments, and projects that we designed for our courses can be easily adapted and adopted in a public cloud or a private cloud to engage students in learning and prepare them for their future careers.

References

[1] P. Abichandani, W. Fligor, and E. Fromm, "A Cloud Enabled Virtual Reality Based Pedagogical Ecosystem for Wind Energy Education," In Proceedings of 2014 IEEE Frontiers in Education Conference (FIE 2014), pp. 1–7, October 2014.

[2] J. Eckroth, “Teaching big data with a virtual cluster,” In Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE '16), pp. 175–180, March 2016.

[3] E. P. Holden, J. W. Kang, D. P. Bills, and M. Ilyassov, “Database in the cloud: a work in progress,” In Proceedings of the 10th ACM conference on SIG-information technology education (SIGITE '09), pp. 138–143, October 2009.

[4] E. Osipov and A. Mousavi, “How to Make a Distributed Programming Course a Big Fun,” In Proceedings of 2014 IEEE Frontiers in Education Conference (FIE 2014), pp. 1–6, October 2014.

[5] R. Pastor, R. Hernandez, S. Ros, D. Sanchez, A. Caminero, A. Robles, L. Tobarra, M. Castro, G. Diaz, E. Sancristobal, and M. Tawfik, “Online laboratories as a cloud service developed by students,” In Proceedings of 2013 IEEE Frontiers in Education Conference (FIE 2013), pp. 1081–1086, October 2013.

[6] A. Rabkin, C. Reiss, R. Katz, and D. Patterson, “Using clouds for MapReduce measurement assignments,” ACM Trans. on Computing Education, vol. 13, no. 1, pp. 2:1–2:18, January 2013.

[7] L. Riliskis and E. Osipov, “Coexistence of cloud technology and IT infrastructure in higher education,” In Proceedings of 2013 IEEE Frontiers in Education Conference (FIE 2013), pp. 805–807, October 2013.

[8] L. F. Z. Rivera, M. M. Larrondo-Petrie, and L. R. D. Silva, “Implementation of cloud-based smart adaptive remote laboratories for education,” In Proceedings of 2017 IEEE Frontiers in Education Conference (FIE 2017), pp. 1–5, October 2017.

[9] K. Salah, “Harnessing the Cloud for Teaching Cybersecurity,” In Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14), pp. 529–534, March 2014.

[10] J. D. Segrelles and G. Molto, “Assessment of cloud-based computational environments for higher education,” In Proceedings of 2016 IEEE Frontiers in Education Conference (FIE 2016), pp. 1–9, October 2016.

[11] E. Shoop, R. Brown, E. Biggers, M. Kane, D. Lin, and M. Warner, “Virtual clusters for parallel and distributed education,” In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE ’12), pp. 517–522, February 2012.

[12] R. S. Weiss, S. Boesen, J. F. Sullivan, M. E. Locasto, J. Mache, and E. Nilsen, “Teaching Cybersecurity Analysis Skills in the Cloud,” In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15), pp. 332–337, March 2015.

[13] A. Wildgoose and S. Bakrania, “Development and implementation of rapid feedback using a cloud-based assessment tool,” In Proceedings of 2017 IEEE Frontiers in Education Conference (FIE 2017), pp. 1–6, October 2017.

[14] W. Zhu, “Hands-On Network Programming Projects in the Cloud,” In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15), pp. 326–331, March 2015.