DUE DATE: **Before end of your lab session**

CSE-381: Systems 2
Exercise #13

Max Points: 20

You should save/rename this document using the naming convention MUid.docx (example: `raodm.docx`).

Objective: The objective of this exercise is to:

1. Experiment with buffer overflow
2. Experiment with denial of service (DOS) attack
3. Experiment with phishing via SMTP protocol

Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from the terminal/output window into this document. You may discuss the questions with your instructor.

Name: **Noah Dunn**

Wait for your instructor to quickly review the following topics from lecture slides prior to proceeding with this exercise:

1. Buffer overflow
2. Denial of Service (DOS) attack
3. Phishing via SMTP

Part #1: Exploring buffer overflow

Estimated time: 20 minutes

Background: In C/C++ programs checking of array bounds are not enabled by default to ensure maximum performance and energy efficiency. However, lack of bounds checks (combined with sloppy programming) can permit access to memory locations beyond the bounds (or size/length) of the array. This is called "buffer overflow" and can be a source of serious security issues in programs, particularly if the programmer has not paid attention to sanitizing user inputs.

Exercise: Explore the issue of buffer overflows via the following procedure:

1. Download the supplied `buffer_overflow.cpp`

2. From a terminal (don't use `NetBeans`), compile the sample code for this part of the exercise using the following commands. **Note**: we are intentionally compiling from terminal to avoid using several compiler flags (that `NetBeans` would add) to ensure we avoid the checks by compiler so we can experiment with buffer overflow:

```
$ g++ -std=c++14 buffer_overflow.cpp -o buffer_overflow
```

3. Study the supplied source code while paying attention to the following aspects:
   a. Note how the `password` input array has been set to 15 characters.
   b. Notice how a `valid` flag (that indicates if password is valid) is set right after the `password`.
   c. What happens if the user enters more than 15 characters? The input overflows the `password` array, *i.e.*, resulting in a buffer overflow, thereby overwriting the default value of `valid` flag.
4. Now run the program and supply the following input that does not cause a buffer overflow because input is less than 15 characters (user input is shown in red):

```
$ ./buffer_overflow
Enter secret code: pa$$word
Authentication failure
```

5. Next run the program and supply the following input that does not cause a buffer overflow because input is less than 15 characters (user input is shown in red):

```
$ ./buffer_overflow
Enter secret code: R0oT!23$5
You are root!
```

6. Now approach the problem as a malicious user. You know the `password` buffer is only 15 characters in size. So enter 16 random characters (yes, it is invalid password, but…) and observe buffer overflow occur (user input is shown in red):

```
$ ./bufer_overflow
Enter secret code: 1234567890123456
Success. You are root!
```

   Note that in the above input, the last character `'6'` overflows the buffer and is written onto the `valid` flag changing its value from false (zero) to true (non-zero) value causing the program to think you have entered a valid password! This type of an attack is called buffer overflow and is a common particularly in C programs.

Avoiding buffer overflow issues

Using suitable data types and sanitizing user inputs can avoid buffer overflow errors. In C++ you avoid such issues by:

❶ Using `std::string` or `std::vector` instead of fixed size arrays

❷ In this course we have been using `NetBeans` with added checks to enforce array bounds checks via compiler flags `-fsanitize=address`. So buffer overflows or invalid pointer operations generate runtime exceptions/errors. Of course, the `CODE` plug-in also uses this flag to ensure that programs do not try malicious operations.

Part #2: Distributed Denial of Service (DDOS)

Estimated time: 20 minutes

Background: Denial of Service (DOS) attack is a mechanism in which the adversary overwhelms the resources of a server, preventing other legitimate users from accessing the server. If may users (typically from different computers) attack the same server then it is considered a Distributed Denial of Service (DDOS).

Exercise: In this exercise you are expected to setup a DOS attack:

1. Download the supplied `dos_attack.cpp` file to your local lab computer.

2. Briefly study the program to observe the following:
   a. Understand the command-line arguments to be supplied to the program
   b. Observe that the program starts several threads
   c. Each thread repeatedly connects to the server and does nothing other than holding on the connection for a few seconds. Think of this as calling a person over the telephone but not having a conversation for a few seconds, which causes the line to be busy for other people who are trying to reach this person. Do this often enough and the person is practically unreachable.

3. Compile it with the following command-line:
```
$ g++ -g -std=c++14 dos_attack.cpp -o dos_attack -lboost_system -lpthread
```

4. Wait for your instructor to run a simple webserver that you can attack. Your instructor will run the web server as shown below and share the `HostName` and `Port` number for the server you should be attacking.
```
$ ./run_server.sh 8080
```

5. Next try to access the web site from a web-browser via the URL: http://HostName:Port/. If none one has already started a DOS-attack, then you should get a simple page.

6. Next start a DOS attack on the server via (you can stop the attack by pressing CONTROL+C):
```
$ ./dos_attack HostName Port 20
```

7. While the attack is running, try to access the web site from a web-browser via the URL http://HostName:Port/ and you will notice that you are only able to access the site very infrequently, if at all. This will be the case when several students are attacking the server and all connections to the server are busy serving requests.

Based on your experience with the DDOS attack, provide brief (2 to 3 sentences) response to the following questions:

1. In your own words, describe how the DDOS attack that you launched actually worked?
   Our Dos-Attack client spun a large amount of threads that consumed all the existing server resources. Because of this, an additional client (Chrome) was not able to connect, as all the resources were hogged because of our malicious client

2. Briefly describe how you could collaborate with your classmates to launch an DDOS attack against some organization?
   If we got a large number of people to run the same program on the same port at the same time, we would orient a DDOS attack.

3. Briefly describe any ideas/guesses that you have about you could design software to defend against such DDOS attacks
   You would want to write a program into your server/ an additional script that checks for excessive connections from a particular IP address and then blocks an IP from connecting when they break that particular rule

Part #2: Sending email via SMTP (to yourself)

Estimated time: 15 minutes

Background: Similar to HTTP sending emails is accomplished using a simple text protocol called Simple Mail Transfer Protocol (SMTP). Similar to all the protocols, SMTP is a text protocol that requires specific commands with suitable format and data.

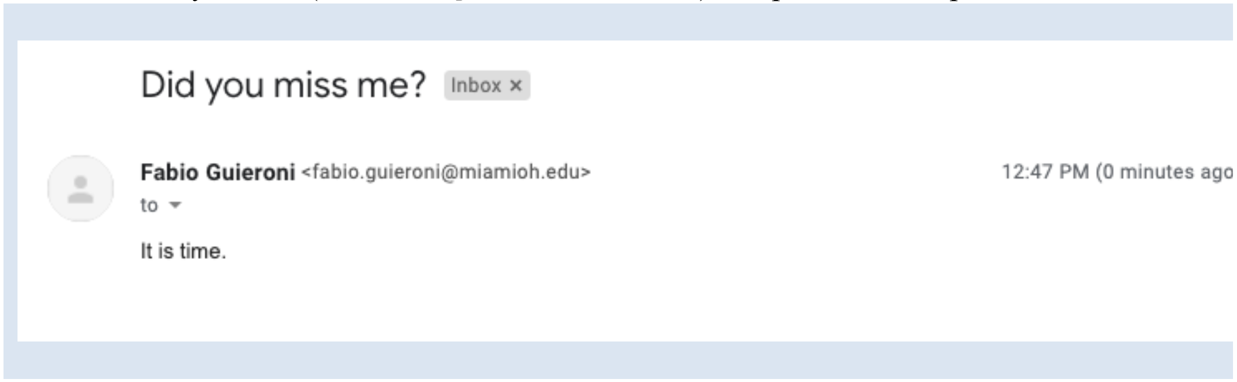
Exercise: This exercise requires you to send yourself an email via SMTP using the following procedure:

1. From a Terminal `ssh` into `os1.csi.miamioh.edu`.
2. Initiate a session with Miami University's mail forwarding server (on port 25) using `telnet` as shown below:

```
$ telnet mailfwd.miamioh.edu 25
```

3. Using the commands from the lecture slide send yourself an email (at your Miami University account) using SMTP protocol with the following information:
   a. `Subject: Test email`
   b. Message -- `This is a test email sent via SMTP`.

4. You should receive the email at your Miami University account. Make a screenshot of the email you sent (as seen in `gmail` mail reader) and past it in the space below:



Did you miss me? Inbox ×

Fabio Guieroni <fabio.guieroni@miamioh.edu>
to

12:47 PM (0 minutes ago

It is time.

Part #3: Phishing via email

Estimated time: 12 minutes

Background: Phishing is the attempt to obtain information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), by masquerading as a trustworthy entity. Phishing via email is often accomplished by masquerading email from a trustworthy authority. SMTP inherently does not have an effective approach for validating sender's email address -- but there are a few precautions that mail server provide (but they are not 100%)

Exercise: In this exercise you are expected to send an email to your neighbor in the lab using SMTP protocol. Most of the procedure is similar to the previous part:

1. From a Terminal `ssh` into `os1.csi.miamioh.edu`.
2. Ask your neighbor's email address and inform them you are planning to send them a phishing email.
3. Initiate a session with Miami University's mail forwarding server (on port 25) using `telnet` as shown below:

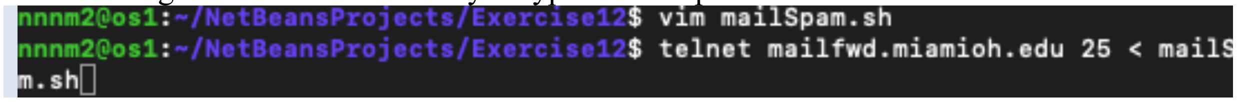
```
$ telnet mailfwd.miamioh.edu 25
```

4. In your SMTP commands change the `MAIL FROM` option to:

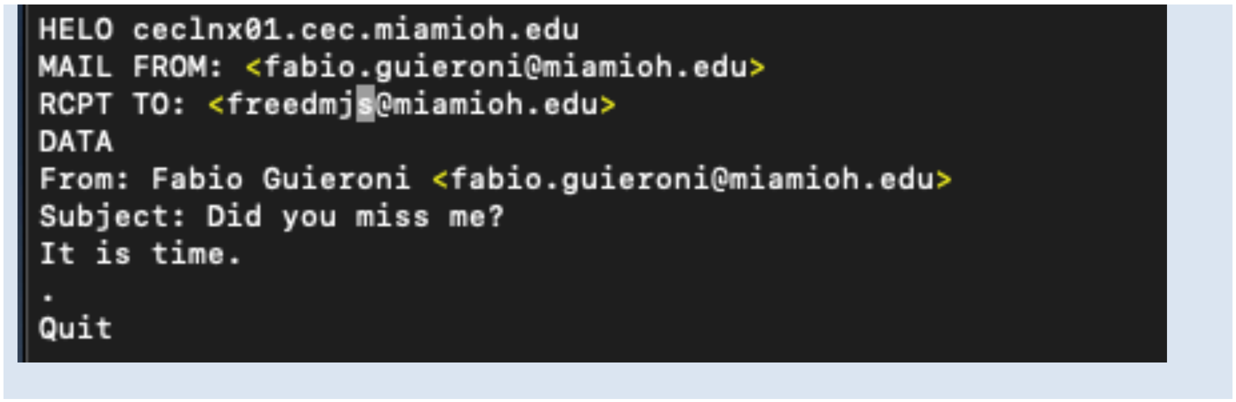
```
MAIL FROM: <registrar@miamioh.edu>
```

5. In your SMTP commands change the `RCPT TO` option to your neighbor's email address.
6. Ensure the subject and email contents is set to:
   a. `Subject: Important information`
   b. Message -- `Our records indicate you are well on your way in CSE-381. Ensure you give your instructor a pie for Thanksgiving.`

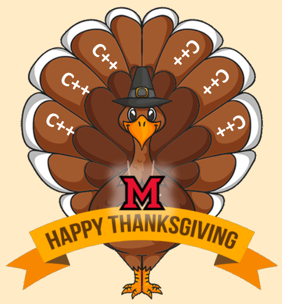
5. Once your email has been accepted for delivery, make a screenshot of the terminal showing the SMTP commands you typed in the space below:



```
nnnm2@os1:~/NetBeansProjects/Exercise12$ vim mailSpam.sh
nnnm2@os1:~/NetBeansProjects/Exercise12$ telnet mailfwd.miamioh.edu 25 < mailS
m.sh
```



```
HELO ceclnx01.cec.miamioh.edu
MAIL FROM: <fabio.guieroni@miamioh.edu>
RCPT TO: <freedmjs@miamioh.edu>
DATA
From: Fabio Guieroni <fabio.guieroni@miamioh.edu>
Subject: Did you miss me?
It is time.
.
Quit
```



Happy Thanksgiving! Please convey my best wishes your family and let them know I am very thankful for working with wonderful people like you. Enjoy time with your family and see you all in December.

Submit to Canvas

Once you successfully completed the aforementioned exercises upload the following files to Canvas.

i. This MS-Word document (duly filled-in) saved as a PDF document.

Ensure you actually **submit** the files after uploading them to Canvas.