Total points: 80 (+5 bonus)

COMP3632 (16-17 Spring)
Assignment 1

Due: 3:00 PM, 10th March, 2017 (Fri)
Milestone Due: 3:00 PM, 24th February, 2017 (Fri)

For the milestone, you only need to submit part (a) of the Programming Assignment; you do not need to submit the Written Assignment yet.

Written assignment

1. [10 points] OnePass offers a password management service. Users of the service can store all of their passwords securely on the OnePass servers for $5 a month. They can have easy access to all of their own passwords by entering a master password, which is never stored on the OnePass servers. However, one possible vulnerability is that attackers can access any user's password if they can guess the master password. OnePass engineers estimate that every year, 10% of their 10,000 user accounts will be hacked by a password-guessing attacker. The problem can be eliminated by enabling two-factor authentication. It will cost the company an extra $3,000 a year to run a server to support two-factor authentication.
   (a) [5 points] For the above scenario, state the System, Asset, Vulnerability, Attack, and Defense.
   (b) [3 points] Assume that the only financial damage caused by the above account-hacking attack is that the hacked user has a 10% chance of noticing right after the hack, after which the hacked user will terminate the service and stop paying the fee. Using Quantitative Risk Analysis, determine if it is worth enabling two-factor authentication.
   (c) [2 points] Suggest one other type of financial damage that may affect the company, besides hacked users terminating the service.

2. [15 points] Read each of the following news stories about malware:
   (a) A new type of Sundown malware is found which makes the victim's computer mine Bitcoins for the attacker. Mining Bitcoins is CPU-intensive, and it is a way to make money for the attacker. The malware is delivered from Javascript code, which can be hosted on an innocuous-looking web site that the attacker invites the victim to access.
   (b) A massive DDoS attack is launched using the Mirai malware against the blog of cybersecurity journalist Brian Krebs. The Mirai malware infected around 50,000 IoT (Internet of Things) devices. It spreads from one device to another by scanning networks for IoT devices and exploits their poor security, such as weak (or non-existent) passwords.

(c) The Angler exploit kit has been shut down after the arrests of several hacking gangs. The kit is often used to collect passwords and credit card information from exploited computers. Angler is capable of hacking legitimate websites and inserting its own code into those sites, thus infecting people who are visiting those web sites.

For **each** of the above news stories, answer the following questions:

i. [9 points] Which of the CIA principles is being violated? Explain why.

ii. [6 points] Classify the malware by stating which class it belongs to, and explain.

3. [15 points] For each of the following Saltzer and Schroeder's Principles of Secure Design:

(a) Economy of Mechanism

(b) Least Privilege

(c) Separation of Privileges

(d) Fail-safe defaults

(e) Open Design

Answer the following questions:

i. [15 points] Explain the principle.

ii. [5 points (bonus)] Give an example from a movie where the principle was violated, or where the principle was useful. The asset being protected does not have to be computer-related. Because it is readily available online, you are not allowed to use the original *Star Wars* trilogy.

Keep in mind that you should use your own words; do not copy and paste from anywhere.

Programming assignment

Viruses [40 points]

Viruses are a type of malware that self-replicate across the victim's files, allowing itself to infect new victims whenever any of those files are sent. In this programming assignment, we will attempt to write a self-replicating program that finds and infects files in the same folder.

(a) [5 points] Write a program called `virus1.cpp` which replicates itself into a target file, for example `victim.cpp`. `virus1.cpp` can rely on a file called `helper.txt`. You will define the contents of `helper.txt` yourself. After compiling `virus1.cpp`, replication is done by running:

`./virus1 victim.cpp`

After that, `victim.cpp` should be exactly the same as `virus1.cpp`. `virus1` should not change `virus1.cpp`.

I will remove `virus1.cpp` after compiling it to test your code.

(b) [20 points] Write a program called `virus2.cpp` which replicates itself into a target file, for example `victim.cpp`. `virus2.cpp` **cannot** rely on any other files. After running

`./virus2 victim.cpp`

`victim.cpp` should be exactly the same as `virus2.cpp`, without changing `virus2.cpp`. Furthermore, `virus2.cpp` must contain your student ID somewhere in its own code (for example, in a comment). The student ID must be copied into `victim.cpp` as well.

I will remove `virus2.cpp` after compiling it to test your code.

(c) [12 points] Write a program called `virus3.cpp` which appends itself to any target `.cpp` file, so that it can be run. `virus3.cpp` takes one input file, for example:

`./virus3 victim.cpp`

After running the virus, `victim.cpp` should still be able to compile successfully, and should maintain its previous functionality, with the added functionality that it can infect files exactly like `virus3`. In other words,

`./victim victim2.cpp`

would also cause `victim2.cpp` to be infected in exactly the same way running `virus3` on it would. You may assume that the `main()` function of the victim file is written as follows:

```
#include <stdio.h>
...
int main(int argc, char** argv) {
        ...
        return 0;
}
```

Assignment 1 Due: 3:00 PM, 10th March, 2017 (Fri)

The file ends with a newline. You may also assume it does not terminate early; only the final `return 0;` will terminate the code. Furthermore, the victim file does not use `argv` in any way. `victim.cpp` is provided as a sample of such a file.

You should try to ensure that the target file will still compile successfully. In particular, even if the same file is infected several times, it should still compile; avoid variable re-declaration errors.

I will remove `virus3.cpp` after compiling it to test your code.

(d) [3 points] Discuss how a virus-scanner would detect files infected with your virus, and clean those files.

Notes on self-reproducing code

To understand how to write self-reproducing code, you are encouraged to read Ken Thompson's Turing Award Lecture, "Reflections on Trusting Trust". A copy of the short paper is hosted on Canvas. The short paper describes how a compiler can be written in such a way as to infect whatever code it compiles. Since C compilers are themselves written with C, an infected C compiler could propagate itself to other compilers and thus other code files.

We ask you to write your own virus in this assignment, but technically the program you would write is not a virus, but only self-reproducing code. To be classified as a virus, it must do something malicious, which it does not. However, one can imagine how it could be used maliciously. Our program is rather easy to defeat. If you are interested in how virus-writers protect their own viruses from virus scanners, you should look into metamorphic and polymorphic viruses.

Submission instructions

All submissions should be done through the CASS system. For this assignment, there are two deadlines: an early Milestone deadline and a normal deadline.

- For the early Milestone deadline, you should submit `virus1.cpp` as per part (a) of the Programming Assignment.
- For the normal deadline, you should submit three files: `a1.pdf` (which should contain your answers to the Written Assignment and part (d) of the Programming Assignment), `virus2.cpp` (Programming part (b)), and `virus3.cpp` (Programming part (c)).

Keep in mind that plagiarism is a serious academic offense; you may discuss the assignment, but write your assignment alone and do not show anyone your answers and code.

The submission system will be closed exactly 48 hours after the due date of the assignment. Submissions after then will not be accepted unless you have requested an extension before the due date of the assignment. You will receive no marks if there is no submission within 48 hours after the due date.

Assignment 1 Due: 3:00 PM, 10th March, 2017 (Fri)

How Your Code Will Be Tested

Using part (a) as an example, your code will be tested as follows:

1. Your submitted `virus1.cpp` and `helper.txt` will be copied to a testing folder, which contains some victim file (for example, `victim123.cpp`.).
2. After changing the working directory to the testing folder, `virus1.cpp` will be compiled with the following command: `g++ virus1.cpp -o virus1`.
3. `virus1.cpp` will be deleted from the testing folder.
4. `virus1` will be run with the following command: `./virus1 victim123.cpp`.
5. `virus1.cpp` will be copied back to the testing folder, and `victim123.cpp` will be compared with `virus1.cpp` using: `diff virus1.cpp victim123.cpp`.

`diff` is a UNIX utility used to find the differences between two files. Ideally, there should be no difference between `virus1.cpp` and `victim123.cpp` after the above steps. (I will accept truly minor compatibility-related differences, such as one extra carriage return or newline at the end of the file.)

For part (c) the target file `victim123.cpp` will also be compiled to test if it is able to infect further files.

Assignment 1 Due: 3:00 PM, 10th March, 2017 (Fri)

Command Line Arguments

C++ (and most other languages) can accept *command line arguments*, which are additional commands given while running the code. Suppose we compiled `mycode` from `mycode.cpp`. If we run the binary code file `mycode` as such:

```
./mycode abc 3 1
```

We say that `mycode` is run with 3 command line arguments; the first one is `abc`, the second one is `3`, and the third one is `1`.

In `mycode.cpp`, the main function header will be written as follows:

```
int main(int argc, char ** argv) {
        ...
}
```

In this code, `argc` is an integer that counts the number of arguments (plus one, because `mycode` also counts), and `argv` is a list of character arrays that contains the arguments. For example:

```
int main(int argc, char ** argv) {
        printf("Number of arguments is: %d, first argument is %s\n", argc, argv[1]);
        return 0;
}
```

The output will be:

```
Number of arguments is 4, first argument is abc
```