**CS 3750 [Fall 2018]**

**Assignment 4**

**Due date: Wednesday, 17 October 2018, by 11:00 p.m.**

***Print your name:* Johnnie Oldfield**

**Answer the following questions:**

1. *BitLocker* is a program that supports encryption of file and directories using a file encryption system and full-disk encryption using another encryption standard. It is suggested that when using BitLocker on a laptop, the laptop should not use standby (sleep) mode, rather should be in hibernate mode. Explain why. [3 points]

In sleep the current state is kept in RAM. In Hibernate it puts all contents of its memory on the hard disk. RAM information could be taken through a cold boot attack, so the data should be more secure on the hard disk while in hibernate mode.

\*In sleep mode is doesn’t copy memory image on the file. BitLocker is encrypting entire disk. In standby mode it is not secure.

1. Consider an access control policy consisting of the following two rules:

Rule1: *A subject S can write to an object O only if security\_level(O) ≤ security\_level(S)*

Rule 2: *If a subject S reads an object O, then the security level of S is updated after the read as MIN{security\_level(S), security\_level(O)}*

Explain whether the above policy prevents integrity violations in a system consisting of some subjects and objects, where all subjects and objects have been assigned some initial security levels based on a hierarchical security-level system. [4 points]

Rule one prevents lower levels writing up which will prevent incorrect data. Rule two states that anyone who rely on lower level data should have their level lowered which will negatively impact integrity.

\*Ignore two. Doesn’t matter. Rule one does not affect integrity.

1. Consider the following C-code fragment:

int main(int argc, char \*argv[]){

char passwd[8];

int continue = 0;

strcpy(passwd, argv[1]);

if(strcmp(passwd, “CS3750”) == 0)

continue = 1;

if(continue)

login(); /\* a method that logs in the user \*/

}

**Note**: *In C, there is no ‘string’ data type, rather a string is viewed as an array of characters. The symbol \* indicates a pointer, which behaves similar to a reference in Java. Therefore, char \*argv[] indicates argv as an array of pointers (references to memory addresses), where each pointer refers to a string. That is, argv is an array of strings where each string item is a command-line argument passed during execution.*

(i) Explain how an attacker can achieve buffer-overflow attack with reference to the variables passwd[] and continue. [4 points]

By exceeding 8 characters with passwd[] it will cause overflow since there is no check. This extra data can have malicious side effects such as messing with continue to make it a non-negative number thus allowing a log in.

(ii) Explain the ideal ordering of the memory locations (assuming memory addresses increases from left to right) that correspond to these two variables so that this attack can be avoided. [3 points]

If the memory locations move from left to right then keeping continue before passwd[] should stop it from being overwritten by overflow.

4. Explain how the special permission bit **setuid** poses a security risk, while the **sticky bit** can be viewed as a protection mechanism. [3 + 3 = 6 points]

Setuid allows the program to run as owner no matter who executes it. So the program will have more privileges than it would normally have. These extra privileges could cause a security issue. They could also find a way to keep these permissions. Sticky bit stops users from deleting/renaming files unless they own them. This will only make things safer.

**Submission instructions:** write your name at the top and include answer to each question on this document, preferably after each question. Please do not write your answers on a separate document or file. Submit the file through BlazeVIEW dropbox.