Homework 05

Due: 11:59 pm, Monday April 16

Please handin your homework assignment on gradescope as a PDF file with each problem on a separate page.

OS Security

Problem 1

You've just received less-than-desirable grades for your previous semester's work at SPECTRE University. Upset, you managed to obtain read permissions for the source code to a critical setuid binary in the University's systems. Aiming to free yourself from the constraint of grades, you peek at the last copy of the code (`delete_user.c`) to see what mischief you can accomplish. It doesn't look like it's the latest version, but it should still give you what you need. See Figure 1 for the source.

You've received word that there is a grade server running on the machines. How might you utilize this executable to kill processes on the machine?

Malware

Problem 2

a) Refer to slide 15 of the lecture on malware, where a virus is abstractly modeled as a program that eventually executes operation infect. Explain why one cannot determine whether a program is a virus simply by checking whether operation infect is present in the code for the program.

b) Suppose you want to use an Internet cafe to login to your personal account on a bank web site, but you suspect that the computers in this cafe are infected with software keyloggers. Assuming that you can have both a web browser window and a text editing window open at the same time, describe a scheme that allows you to type in your userID and password so that a keylogger, used in isolation of any screen captures or mouse event captures, would not be able to discover your userID and password.

c) Typical anti-malware software works by scanning the directories in the file system for any suspicious programs. When scanning a directory, the anti-malware scanner will call a system function that lists the contents of the directory. The program then reads each file in the directory, looking for any malicious content. Describe two ways a rootkit could hide itself from this sort of malware scanner.

d) To overcome the problems with the above detection, an alternate approach to detect rootkits is to reimplement the open() and read() system calls and look at the file system directly. That is, the program would communicate with the hardrive device driver [1] directly to get a picture of what's really going on. How could a rootkit author circumvent this detection technique? Why might malware authors be reluctant to employ such a technique?

[1] A device driver is a piece of software in an operating system that is used to interact with hardware devices. The driver is created by the hardware vendor (e.g. Seagate) and it provides a layer of abstraction that makes it easier to interact with their device (e.g. a hard drive).

```c
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

// Validate that the user with the given uid
// has the given password hash.

// Stub; to be replaced with real implementation later.
bool validate_password(int uid, char *hash) { return false; }

// hasher is the type of functions which
// can compute hashes. The second argument
// is a byte array of the given length, and
// the third argument is a pointer to a buffer
// of sufficient length to hold the hash itself.
typedef void (*hasher)(int len, char *input, char *hash);

// find_hash_by_name returns a hash function
// which can compute the named hash, or NULL
// if no such function is found.
hasher find_hash_by_name(char *name);

// Stub; to be replaced with real implementation later.
hasher find_hash_by_name(char *name) { return NULL; }

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <uid> <password>\n", argv[0]);
        exit(1);
    }

    int uid = atoi(argv[1]);
    char password[128];
    char hash[20]; // SHA-1 hashes are 160 bits = 20 bytes
    hasher hash_function = find_hash_by_name("SHA-1");
    strcpy(password, argv[2]);
    if (hash_function == NULL) {
        fprintf(stderr, "could not find hash!\n");
        exit(1);
    }

    hash_function(strlen(password), password, hash);
    if (!validate_password(uid, password)) {
        fprintf(stderr, "could not validate password.\n");
        exit(2);
    }
    // TODO: Actually perform the deletion.
    printf("deleting...\n");
}
```

Figure 1: Code for the setuid program `delete_user.c` in Problem 1.

Storage Encryption

Problem 3

Answer the following questions and justify your answers:

a) Consider an Apple macOS laptop employing FileVault 2 to encrypt the drive. Is turning off the laptop (instead of placing it in sleep mode) and waiting a few minutes before leaving the laptop unattended an effective defense against cold boot attacks?

b) Consider now a Microsoft Windows laptop employing Bitlocker to encrypt the drive, where the volume master key (VMK) for the drive is kept in the TPM and released to the operating system upon verification that the operating system has not been tampered with. Is turning off the laptop (instead of placing it in sleep mode) and waiting a few minutes before leaving the laptop unattended an effective defense against cold boot attacks?

c) Identify two distinct precautions other than constantly maintaining physical possession that could be used by the owner of a laptop with full disk encryption to defend against evil room cleaner attacks. For each precaution, discuss its security and usability.

SSL/TLS

Problem 4

Websites that prefer using HTTPS will often include in the HTTP version of the site an HTTP or JavaScript redirect to the HTTPS version of the site so that the client's browser will upgrade the connection to use HTTPS. Explain how an active man-in-the-middle attacker can prevent the upgrade to HTTPS from ever occurring.

Network Security

Problem 5

Returning from your spring break trip, you have a very long layover at the airport and so you decide to start the new CS166 homework. The airport is greedy, and only allows you to use WiFi for 5 minutes without paying. You connect to the airport WiFi but after 5 minutes you are told that your time has expired and you must pay to continue using it. But you dont want to spend any money because you already spent all of your money buying novelty sweaters for your pet lizard. You try several methods for getting around the paywall, such as clearing your cookies, using a different browser, and reconnecting multiple times with different IP addresses. Alas, every time you are still told your access has expired. You do have a small success when you try with a different device (your phone): you get another five minutes out of it, but then you run into the same issue there too.

Youre getting impatient, and youre tired from carrying around all your reptilian apparel. In a moment of CS166-inspired brilliance, you figure out a solution to the problem.

a) How is the airport tracking you? How do you know this?

b) How can you circumvent their watchful wily ways by deploying a simple solution on your laptop every 5 minutes?

c) Having to run a solution every 5 minutes gets quite annoying after a while, and you decide to look for a better solution. You notice that every once in a while, someone nearby will take out their wallet and enter their credit card information to pay for unlimited WiFi. Sometimes they wont even use the WiFi

very long—they'll get on their plane within an hour or so. What a terrible waste of unlimited WiFi! With this knowledge, how could you get unlimited WiFi without having to pay for it yourself? Note: the solution is *not* to steal their credit card information.

d) Could your home Internet service provider (e.g., AT&T, Comcast, Cox, etc.) track you in the same way as the airport WiFi network? Why or why not?