Lab Exercise 4

Due: November 18, 2016 **Points:** 100

For this laboratory exercise, you are to work individually.

This problem asks you to write a program using C or C++ that will introduce you to come complexities of managing privileges, as well as obstacles you might encounter when doing computer security work.

A computer science student is helping out on a project that involves monitoring a network. The student's job is to write a program that will pluck the URLs for all HTTP traffic from the network. This requires *root* privileges, but for policy reasons the professor who is running the project cannot give the student those privileges. The professor's solution is to create a small program, called *runpriv*, that will make a second program called *sniff*, that the student will write, setuid to *root*. This way, the student can write the program to get the URLs from the network and run it without having *root* permissions and without asking a system administrator to make the program privileged at each iteration.

The program *runpriv* works as follows:

1. Check that the student is running the program by comparing the real UID of the process with that of the student. (Assume you are the student for this testing.) If the test fails, print an error message and exit.
2. Prompt the user for his or her password, and validate it against the authentication credential in the UC Davis Central Authentication System (use the program *kinit*(1) for this). If the password entered is incorrect, print an error message and exit.
3. If the current working directory does not contain a file called *sniff*, print an error message and exit.
4. If *sniff* is not owned by the student, or is not executable by the owner of the file, or can be read, written, or executed by anyone else (except, of course, *root*), print an error message and exit. This step checks that the student owns the file; that the student can execute it; and that no-one else has any rights over it.
5. If *sniff* was created or modified over 1 minute ago, print an error message and exit.
6. Change the ownership of *sniff* to *root* (UID 0), its group to *proj* (GID 95), and its protection mode to 4550 (meaning setuid to owner, and only readable and executable by the owner and group members — when you call *chmod*(2), you *must* have the leading "0", or you will get strange results. For this exercise, you must use the *chown*(1) program to change the owner and group. This means you must execute that program from *runpriv*, giving the appropriate arguments. (In the CSIF, the command will fail because there is no group "proj" — just let the error message print, and continue.)

Your job is to write *runpriv*.

Your program must be robust! Out of the 100 points for this program, 40 will come from the robustness and security protections you add to it to keep it from being abused.

Submitting Your Program

You must submit either a tar archive or a compressed tar archive to Canvas, as described in the handout **All About Programs**. Do this as follows:

1. Create a directory called lab4-*yourlastname*, where *yourlastname* is your last name.
2. Copy the source code (*not* the executable!) into that directory.
3. Create a Makefile in that directory. When we test your program, we will change to the directory and type "make". So be sure your Makefile correctly compiles your program on the CSIF!
4. Now create your documentation — for this program, a README saying how to compile it, and what it does, is sufficient.
5. Then create either a tar archive (the archive's name is to end in ".tar") or a compressed tar archive (the compressed archive's name is to end in ".tgz"), and submit that to Canvas.

That's it!

Regrading

When we grade your program, 40% of the grade will be based on robustness, which includes handling errors and problems gracefully as well as good programming style. If you lose points because of this, we will give you a week to modify your program and resubmit it. We will then regrade *only* the robustness, and add back 75% of the points you regain. So if your score on the robustness part is 20 out of 40, you can get up to 15 of the other 20 points back by fixing your program and resubmitting it.