Lab Exercise 4

Due: June 6, 2018 **Points:** 100

For this laboratory exercise, you are to work in teams of 2–3 people. When you turn in your results, also upload a README file giving the names and UC Davis email addresses of all members of your group. Only one person needs to upload the program; the other members should upload a file named README identifying the other members of their group, as above, and note who uploaded the answers.

This problem asks you to write a program will introduce you to some 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 execute 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, but make the UID be a macro called **`STUDENT_UID`**.) 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; run it as a subprocess). 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 GID 95, and its protection mode to 4550 (meaning setuid to owner, and only readable and executable by the owner and group members). Do this as follows. Use the *chown*(2) system call to change the owner and group, and then the *chmod*(2) system call to change the permissions. For the latter, you *must* have the leading "0", or you will get strange results. Note that in the CSIF, the *chown* system call will fail because you cannot give a file away to *root*. Don't worry about that; just print an error message and continue.

Your job is to write *runpriv*.

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!

Your program must be robust! Out of the 100 points for this program, 50 will come from the robustness and security protections you add to it to keep it from being abused. Unlike lab exercise 3, you cannot resubmit this one after fixing errors, so write your program carefully.