Program #3: Changing Program Permissions Carefully (Revised)

Due: May 8, 2015 **Points:** 100

This program 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 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.) If the test fails, print an error message and exit.
2. If the current working directory does not contain a file called *sniff*, print an error message and exit.
3. 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.
4. If *sniff* was created or modified over 1 minute ago, print an error message and exit.
5. 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*.

Please submit your program as a "tar" file. Don't forget to include a Makefile that will automatically compile your program. And as always, remember we will execute the program on the CSIF systems, so be sure it works there.

Your program must be robust! One goal of this program is to teach you how to program with security in mind. To that end, we are experimenting with a "secure programming clinic". The purpose of this clinic is to review programs with an eye to helping you understand potential security and robustness problems and working around them.

To obtain help from the clinic, send an email to "progreview@ucdavis.edu". Ask to meet with a member of the clinic (currently it is run by two graduate students, so be patient). Attach a *working version* of your program. A graduate student will respond and arrange a meeting time, and go over your program with you. The meeting may be in person or over Skype or some other remote communication mechanism, depending on schedules. Then you can revise your program as appropriate before you submit it.

We will not deduct any points for your not using the clinic, nor will we give you extra credit for using it. But 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. If you submit your program and lose points for these, we will give you a week from the time we turn back the homework to consult with the clinic and fix the problems. Then we will regrade anything you resubmit. You can get back up to 80% of the points you lost for non-robustness or security problems. But in order to get these back, you *must* consult with the clinic before turning in the corrected program for a regrade!

Extra Credit

Add the following step between steps 1 and 2:

1a. Prompt the user for his or her password, and validate it against the one stored in the password file. If the password entered is incorrect, print an error message and exit.