Set-RandomUID Lab

Lab Description

When we need to run a program that we do not totally trust, we really do not want to run the program in our own account, because this untrusted program might modify our files. It is desirable if the operating system can create a new user id for us, and allows us to run the program using this new user id. Since the new user id does not own any file, the program cannot read/modify any file unless the file is world-readable/writable. We will design such a mechanism for `Minix` in this lab.

Lab Tasks

In this lab, you need to design and implement a `Set-RandomUID` mechanism. When a `Set-RandomUID` program is executed, the operating system randomly generates a non-existing user id, and runs the program with this new user id as the effective uesr id. You can consider `Set-RandomUID` as an opposite to the `Set-UID` mechanism: `Set-UID` allows users to escalate their privileges, while `Set-RandomUID` allows users to downgrade their privileges. The implementation of `Set-RandomUID` can be similar to that of `Set-UID`. The following list provides some useful hints:

1. To mark a program as a `Set-RandomUID` program, we can use the unused *sticky* bit in the permission field of the I-node data structure (defined in `/usr/src/fs/inode.h`). You might need to modify the `chmod.c` file under the `/usr/src/commands/simple` directory.

2. Before a program is executed, the program will be loaded into memory and a process will be created. The system call `exec` in `/usr/src/mm/exec.c` is invoked to handle the tasks. You might need to modify this file.

3. There are a number of potential loopholes in the `Set-RandomUID` mechanism if you do not take care of them in your design. In your lab report, you need to explain whehther they are loopholes. If yes, you need to fix the loopholes in your implementation, and also explain your solutions in your lab report.

   (a) Is it possible for a malicious program to use `setuid()` and `setgid()` system calls to defeat `Set-RandomUID`?

   (b) Is it possible for a malicious program to defeat `Set-RandomUID` by creating new processes?

4. Bob decides to reserve 0 to 999 for the IDs of actual users. Therefore random user ID starts from 1000, so Bob writes the following statement to generate a random ID.: "`unsigned int randomID = rand() + 1000;`" Then he assigns the `randomID` to the effective user ID of the process. Can anything go wrong because of this statement? Please explain.

5. There might be other potential loopholes. We will award up to 50 bonus points to the identified loopholes, 10 points for each.

Submission and Demonstration

You should submit a detailed lab report to describe your design and implementation. You should also describe how you test the functionalities and security of your system. You also need to demonstrate your system to the TA. Please sign up a demonstration time slot with the TA.