TP4

The course homepage is here :

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1415/`.

You will find the slides from the course and some other files for the exercise there.

Details of shell commands and C functions can be obtained by using the `man` command.

1 Linking

Consider the following small fragment of a C program :

```
x=y+z;
printf("x=%d\n",x);
```

The first statement can be directly translated into assembly code that modifies registers and/or memory. The second statement is less obvious ; in fact its output might appear on the screen (in some window), or in a file (with shell-level redirection). Analyzing the assembly code of this fragment reveals that the second statement is translated into a `call` statement to `printf`, which itself is not specified.

`printf` is in fact a function whose code is supplied by a so-called *shared library*. The prevalent compilation model in Unix (and other OS) is *dynamic linking* : The assembly code of a program is stored on the hard disk in an incomplete form ; before it is executed, that assembly code has to be completed with the concrete addresses for functions such as `printf`. This process is called *linking*.

Normally, the system knows where to find the right libraries that it needs. This behaviour can be influenced by two things :

- The environment variable `LD_LIBRARY_PATH` tells the linker in which directories it should look for shared libraries.
- The `LD_PRELOAD` variable specifies a number of shared variables that will be loaded before any others.

The primary purpose of shared libraries is to save memory : many different programs may run at the same time, but they will share the code for system functions like `printf` (and many others !). A secondary effect is that the behaviour of a program can be changed by exchanging one implementation of a shared-library function with another. We will explore some uses of this. A demonstration will be given during the course. After this, you should try the following, using (i) the files available on the course webpage, (ii) the tools shown during the demonstration, and (iii) your own creative uses of `LD_PRELOAD`.

1. Regard the (pre-compiled) program `password1`, which asks you for a password. Find the password.

2. Do the same for `password2`.
3. These two programs have some serious security flaws, as you should have found out. How do think such problems can be avoided ?
4. The program `game` lets you guess a number between 1 and 100. Unfortunately for you, it is chosen randomly in every execution. Find out how you can cheat and win with the first try anyway.
5. Other than cheating or hacking, can you think of legitimate uses of `LD_PRELOAD` or related mechanisms ?