SWE2001 Spring 2014
Programming Assignment 3: A Sandbox To Protect
Assigned: May 29, Due: June 9

1 Introduction

All of a sudden, the nefarious *Dr. Evil* has presented you a program. Although you think that some dangerous booby traps are hidden in the program, you are eager to run the program out of curiosity. You know that curiosity killed the cat. Thus you decided to run it in a sandbox.

Dr. Evil is a very sly person. So, he set up a sandbox neutralizer in his program. If the program detects suspicious activities, the program will stop its execution, and you will never know what the program is for. So your sandbox must provide the illustion that the program is running on a baremetal hardware without a sandbox. This time, your mission is to build a sandbox that can confine the program not to make a mess in your system while you observe its behavior. Good luck, and welcome again to the bomb squad!

Step 1: Get the Program

You can obtain your bomb by pointing your Web browser at:

```
http://csl.skku.edu/uploads/SWE2001S14/evil.tgz
```

The downloaded tarred zip file contains a directory named `evil`. The directory contains the program aloing with a data directory. The program is a x86-32 Linux binary. The data directory `data` includes two important data files, which your sandbox must protect. In other words, the files in the data directory must not be touched by the given program in any cases.

Step 2: Build the Sandbox

Your job for this lab is to build a sandbox. The sandbox must be able to catch all attempts to delete or modify the files in the data directory. The program will work correctly only when the data directory exists in the same directory that the program is located in. The program can be executed by the following command.

```
linux>  ./evil
```

There is a rumor that the program consists of five stages. If your sandbox works correctly, the program from Dr. Evil will successfully go through all five stages without any changes to the data files.

Logistics

This is an individual project. All handins are delivered via e-mail. Clarifications and corrections will be posted on the course message board.

Only the assignments submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.

Handin

You are supposed to hand in the source code of your sandbox, makefile, and documentation of your sandbox. All these files must be in the same directory named after your student ID. For example, if your student ID is 2013310123, the directory must be named 2013310123. Invoking the `make` command in the directory should automatically produce the sandbox, which is probably a shared library. The produced shared library must be named after your student ID, too. For example, `2013310123.so`.

After you finish the implementation of the sandbox, tar and gzip the directory in the upper directory. The tarred-and-gzipped file has the same name, but with the `tgz` extension, like `2013310123.tgz`. Send the `tgz` file via e-mail to *homework.skku@gmail.com*. The title of the e-mail should be *[SWE2001 PA#3] student ID* like *[SWE2001 PA#3] 2013310123*.

Hints

You should clearly understand procedure interpositioning to finish this assignment.

You have no information what procedures to intercept. Disassemble help you to investigate what each stage does. However, it will not tell you what procedures (or functions) each stage calls. `readelf` is a tool that shows the contents of each section in an ELF file. It will be a handy tool for you in hunt for the function names to intercept.