CSCE 3402 Spring 2021
Exercise 6 – Hooking the Fork System Call
Kernel Module to hook the Fork System Call

Assigned: Tuesday, March 16th in Lab

Due: Tuesday, March 23th at Lab time
Delayed submission with penalty until Thursday, March 25th at 11:55pm.

Goals
The goal of this lab exercise is to extend your kernel module you developed in the previous exercises to hook the **fork** system call and keep track of the total number of times it get invoked throughout the kernel up time, starting from the moment your kernel module get loaded. It is very essential to do this step to be able to do the next lab exercise.

Details
This lab exercise is an individual exercise that you need to carryout on your own. After you have managed to get the running kernel version, the address of the system call table, and the address of the fork system call in the previous lab exercise, you should use that to hook the fork system call and redirect the fork to a hook function within your kernel module. You should modify the entry inside the system call table that points to the fork system call to point to your own hook function. You should declare a global variable in your module that should be initialized to zero and get incremented every time your hook function is executed. You should call the original fork system call at the end of your hook function; easy, you already have its address. Finally, print the number that keeps track of the number of forks so far every time it is incremented by 10, and monitor it in the syslog or dmesg. On unloading your module, you should essentially unplug your hook and restore the system to its original setup.

You think it is that easy :) do not bet on it :)

Your problem is that the virtual page that maps the system call table is mapped with read-only privileges. Your main challenge is to update the virtual page mapping to be writable, modify the system call table, and set it to read-only again.

Hint: there is another catch here, do you really think that you got the correct entry for the fork in your previous example??? :) you need to look carefully into that. Please, you need to read documentation a bit.

IMPORTANT Note: Please make sure that you boot up with the kernel grub entry you created in the lab exercise #3 that has the KASLR disabled.

What to submit

1. All the C code you wrote for the version kernel module.
2. You Makefile.
3. A small readme file explaining how to use your make files to compile the programs.

How to submit:

Compress all your work: source code, report, readme file, and any extra information into a zip archive. You should name your archive in the specific format <Student_ID>_<Name>_Lab6.zip. Finally, upload your code to blackboard.

Grade

This Lab exercise is worth 10 % of the overall course grade. The exercise will be graded on a 100% grade scale, and then will be scaled down to the 10% its worth. The grading of the assignment will be broken down as follows:

1. 10 % for just submitting a meaningful assignment before or on the due date. This 10% does not account for the correctness of your assignment but submitting an empty assignment without code will definitely results in loosing this 10% and consequently the whole grade of this assignment.
2. 80 % for the correctness and the quality of the submitted code and make files.
3. 10 % readme file.

Delays

You have up to 2 working days of delay, after which the assignment will not be accepted and your grade in that case will be ZERO. For every day (of the 2 allowed days), a penalty of 10% will be deducted from the grade. And of course, you will lose the 10% mentioned in point 1 above under the “Grade” section.