운영체제실습 보고서

Assignment 2

학 과 : 컴퓨터정보공학부

담당교수 : 김태석 교수님

학 번 : 2018202018

이 름 : 유승재

1. Introduction

이번 과제는 새로운 시스템 콜 ftrace를 시스템 콜 테이블에 등록하고 ftrace를 hijack하여 기존의 ftrace 함수를 pid를 입력 받고 이를 추적하여 그 pid에 해당하는 프로세스의 이름을 출력하는 커널 모듈 ftracehooking과 기존에 등록되어 있는 open, read, write, lseek, close 함수를 hijack하여 새로운 함수로 대체하고 이를 ftracehooking과 연동하여 사용하는 커널 모듈 iotracehooking을 작성하고 출력하여 결과를 확인하는 과제입니다.

2. Result

```
os2018202018@ubuntu:~$ cd Downloads/linux-4.19.67/assignment2/
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$ lsmod |grep tracehooking
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$ sudo insmod ftracehooking.ko
[sudo] password for os2018202018:
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$ sudo insmod iotracehooking.ko
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$ ./test
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$ dmesg
```

Lsmod를 통해 ftracehooking.ko와 iotracehooking.ko가 등록되어 있는지 확인합니다.

그 후 적재되어 있지 않은 것을 확인 후 ftracehooking.ko와 iotracehooking.ko를 적재합니다.

적재 후 open, read, write, lseek, close를 사용하는 test 파일을 실행합니다.

Dmesg를 통해 올바르게 출력되었는지 확인합니다.

```
[   97.449130] OS Assignment2 ftrace [2130] Start
[   97.449135] [2018202018] test file[abc.txt] stats [x] read - 20 / written - 26
[   97.449137] open[1] close[1] read[4] write[5] lseek[9]
[   97.449138] OS Assignment2 ftrace [2130] End
os2018202018@ubuntu:~/Downloads/linux-4.19.67/assignment2$
```

Dmesg의 결과입니다.

2130 = pid를 가진 test 프로세스에서 abc.txt 파일에 read 20bytes, write 26bytes를 하였고 각각의 함수가 몇 번씩 실행되었는지 확인할 수 있습니다.

```
printk("OS Assignment2 ftrace [%d] Start", pid_temp);
printk("[2018202018] %s file[%s] stats [x] read - %ld / written - %ld", task->comm, file_name, read_bytes, write_bytes);
printk("open[%d] close[%d] read[%d] write[%d] lseek[%d]", open_count, close_count, read_count, write_count, lseek_count);
printk("OS Assignment2 ftrace [%d] End\n", pid_temp);
```

해당하는 코드입니다.

```
static asmlinkage int my_ftrace(const struct pt_regs *regs)
{
    pid_t pid = (pid_t)regs->di;
    struct task_struct *task;
    task = current;
```

task_struct 형태로 지정된 task를 current를 통해 현재 프로세스를 가리키게 하고 task->comm을 통해 프로세스의 이름을 출력하였습니다.

```
extern int open_count;
extern int read_count;
extern int write_count;
extern int lseek_count;
extern int close_count;
extern size_t read_bytes;
extern size_t write_bytes;
extern char file_name[100];
```

또한 count, byte 등의 변수 들을 iotracehooking.c에서 EXPORT_SYMBOL을 통해 연동하여 open, read 등의 시스템 콜을 hijack한 커스텀 함수들에서 값을 계산하여 다시 ftracehooking.c에 보내주고 그것을 출력하는 과정을 거쳤습니다.

Test 실행 파일을 생성하는데 사용된 test.c 파일은 조교님이 올려 주신 test.c 파일을 그대로 사용하였습니다.

3. 고찰

이번 과제를 통해 처음으로 커널 모듈 컴파일을 해봤습니다. 이전까지의 일반적인 user mode에서의 함수와는 다르게 함수를 생성하는 과정에서 신경써줘야 하는 부분도 많이 있었고 컴파일이 올바르게 되었다고 하더라도 컴파일된 .ko파일을 모듈에 insmod를 통해 넣는 순간 vmware 자체가 멈춰버리는 문제도 종종 발생하였습니다. 또한 이러한 문제가 발생하는 경우 오류 메시지 자체가 출력되지 않아 어떠한 문제로 다운되었는지 조차 알 수 없어 많은 문제가 있었습니다. 커널 모듈 코딩 과정 또한 익숙하지 않은 함수와 기존에는 문제 없이 작동하던 코드가 커널 모드에서는 작동하지 않는 경우가 너무 많았습니다.

이번 과제는 평소에 자주 접하기 힘든 커널 모듈 구현 과정에 대해서 알게 되었습니다. 자주 접하기 힘든 부분이기 때문에 많은 오류를 보았지만 과제를 진행하면서 커널 모듈에 대해서 이론적으로 공부했을 때보다 더 깊이 이해가 되는 것 같았습니다.

4. Reference

https://stackoverflow.com/questions/59851520/system-call-hooking-example-arguments-are-incorrect - iotracehooking.c 구현 과정의 open 함수 hijack 부분