

# HOOK SYSTEM CALL PNAME

Để bắt đầu, hãy hiểu rằng bây giờ bạn sẽ tạo một kernel module dưới dạng hook, không phải là một system call. Các module có thể được tải và gỡ bỏ khỏi kernel tại bất kỳ điểm nào (với điều kiện bạn được ủy quyền) bằng cách sử dụng các lệnh insmod và rmmod. Để xem tất cả các module hiện đang chạy, bạn sẽ sử dụng lsmod. Module mới của tôi về mặt kỹ thuật nó sẽ hooking vào system call pname mà tôi đã tạo trước đó

Tạo một thư mục bạn chọn để lưu trữ hook và cd vào nó. Của tôi là thư mục root.

## Tìm địa chỉ sys\_call\_table:

`cat /boot/System.map-3.16.36 | grep sys_call_table`

```
root@debian:~/captainHook# cat /boot/System.map-3.16.36 | grep sys_call_table
ffffffff81601680 R sys_call_table
ffffffff8160cb40 R ia32_sys_call_table
root@debian:~/captainHook#
```

Copy địa chỉ nào để paste vào code

captainHook.c:

```
#include <asm/unistd.h>
#include <asm/cacheflush.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <asm/pgtable_types.h>
#include <linux/highmem.h>
#include <linux/fs.h>
#include <linux/sched.h>
#include <linux/moduleparam.h>
#include <linux/unistd.h>
#include <asm/cacheflush.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("D0hnuts");
/*MY sys_call_table address*/
//ffffffff81601680
```

```

void **system_call_table_addr;

/*my custom syscall that takes process name*/
asmlinkage int (*custom_syscall) (char* name);

/*hook*/
asmlinkage int captain_hook(char* play_here) {
    /*do whatever here (print "HAHAHA", reverse their string, etc)
    But for now we will just print to the dmesg log*/
    printk(KERN_INFO "Pname Syscall:HOOK! HOOK! HOOK! HOOK!...ROOOFFIIIOO!");
    return custom_syscall(play_here);
}

/*Make page writeable*/
int make_rw(unsigned long address){

    unsigned int level;
    pte_t *pte = lookup_address(address, &level);
    if(pte->pte & ~_PAGE_RW){
        pte->pte |= _PAGE_RW;
    }
    return 0;
}

/* Make the page write protected */
int make_ro(unsigned long address){

    unsigned int level;
    pte_t *pte = lookup_address(address, &level);
    pte->pte = pte->pte & ~_PAGE_RW;
    return 0;
}

static int __init entry_point(void){

    printk(KERN_INFO "Captain Hook loaded successfully..\n");
    /*MY sys_call_table address*/
    system_call_table_addr = (void*)0xffffffff81601680;

    /* Replace custom syscall with the correct system call name (write,open,etc) to hook*/
    custom_syscall = system_call_table_addr[__NR_pname];

    /*Disable page protection*/
    make_rw((unsigned long)system_call_table_addr);

```

```

/*Change syscall to our syscall function*/
system_call_table_addr[__NR_pname] = captain_hook;
return 0;
}

static int __exit exit_point(void){

    printk(KERN_INFO "Unloaded Captain Hook successfully\n");

    /*Restore original system call */
    system_call_table_addr[__NR_pname] = custom_syscall;

    /*Renable page protection*/
    make_ro((unsigned long)system_call_table_addr);
    return 0;
}

module_init(entry_point);
module_exit(exit_point);

```

## Tạo Makefile:

### [nano Makefile](#)

```

obj-m += captainHook.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

```

```

root@debian:~/captainHook# cat /boot/System.map-3.16.36 | grep sys_call_table
ffffffff81601680 R sys_call_table
ffffffff8160cb40 R ia32_sys_call_table
root@debian:~/captainHook# cat Makefile
obj-m += captainHook.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

root@debian:~/captainHook# ls
captainHook.c  Makefile
root@debian:~/captainHook#

```

## Test hook trước khi tải nó vào kernel:

[Make](#)

```

root@debian:~/captainHook# make
make -C /lib/modules/3.16.36/build M=/root/captainHook modules
make[1]: Entering directory '/usr/src/linux-3.16.36'
  CC [M]  /root/captainHook/captainHook.o
In file included from /root/captainHook/captainHook.c:1:0:
/root/captainHook/captainHook.c: In function '__exittest':
include/linux/init.h:335:4: warning: return from incompatible pointer type
    { return exitfn; }      \
      ^
/root/captainHook/captainHook.c:81:1: note: in expansion of macro 'module_exit'
module_exit(exit_point);
^
  Building modules, stage 2.
MODPOST 1 modules
  CC      /root/captainHook/captainHook.mod.o
  LD [M]  /root/captainHook/captainHook.ko
make[1]: Leaving directory '/usr/src/linux-3.16.36'
root@debian:~/captainHook# ls
captainHook.c  captainHook.mod.c  captainHook.o  modules.order
captainHook.ko  captainHook.mod.o  Makefile      Module.symvers
root@debian:~/captainHook#

```

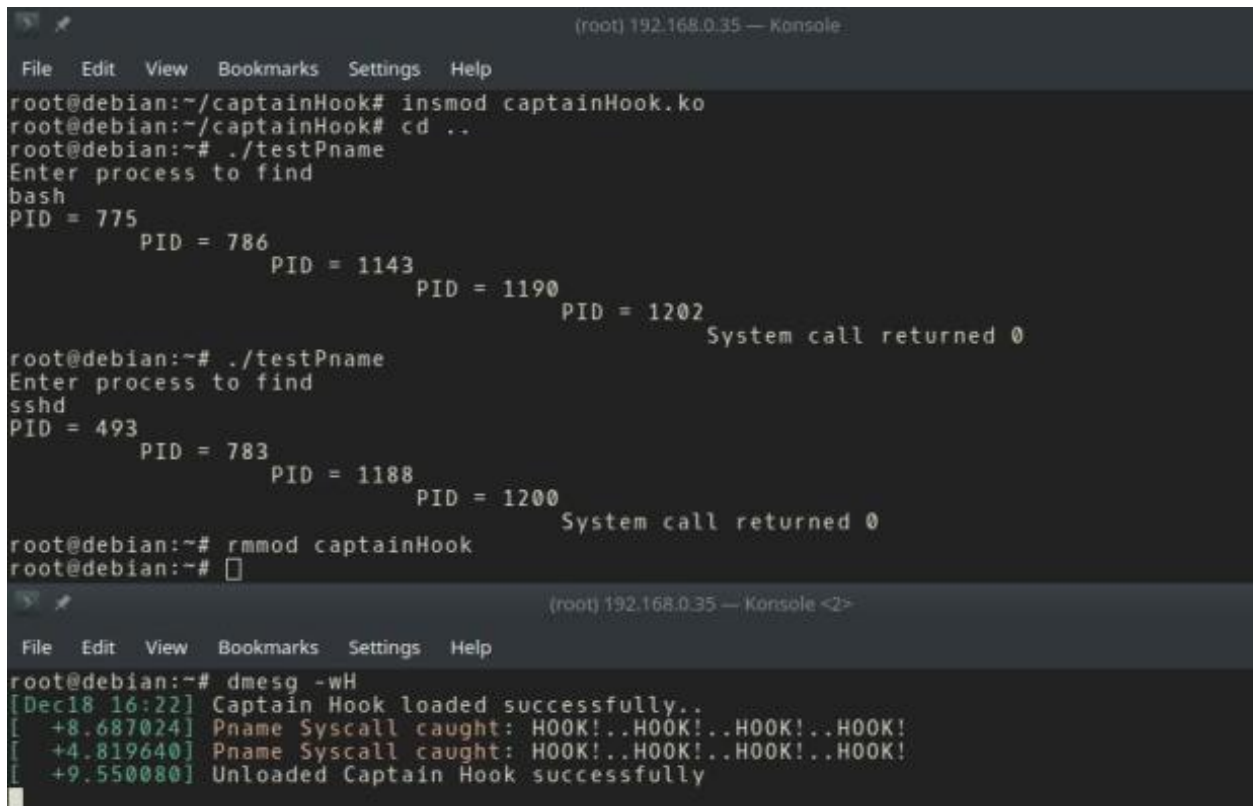
Bây giờ hãy mở một terminal khác, gõ lệnh sau để xóa dmesg và sau đó đọc tail output của nó sau khi bạn insert module và chạy testPname.

### FIRST TERMINAL:

```
DMESG -C  
DMESG -WH
```

### SECOND TERMINAL:

```
INSMOD CAPTAINHOOK.KO  
CD ..  
./TESTPNAME  
RMMOD CAPTAINHOOK
```



The image shows two terminal windows. The top window, titled '(root) 192.168.0.35 — Konsole', shows the process of loading and testing the 'captainHook.ko' module. The user runs 'insmod captainHook.ko', then 'cd ..', and then './testPname'. The test program prints out several PIDs (775, 786, 1143, 1190, 1202) and 'System call returned 0'. The user then runs './testPname' again, which prints out more PIDs (493, 783, 1188, 1200) and 'System call returned 0'. Finally, the user runs 'rmmod captainHook'. The bottom window, titled '(root) 192.168.0.35 — Konsole <2>', shows the output of 'dmesg -WH'. It displays three log messages: 'Captain Hook loaded successfully..', 'Pname Syscall caught: HOOK!..HOOK!..HOOK!..HOOK!', and 'Unloaded Captain Hook successfully'.

```
(root) 192.168.0.35 — Konsole  
File Edit View Bookmarks Settings Help  
root@debian:~/captainHook# insmod captainHook.ko  
root@debian:~/captainHook# cd ..  
root@debian:~# ./testPname  
Enter process to find  
bash  
PID = 775  
    PID = 786  
        PID = 1143  
            PID = 1190  
                PID = 1202  
                    System call returned 0  
root@debian:~# ./testPname  
Enter process to find  
sshd  
PID = 493  
    PID = 783  
        PID = 1188  
            PID = 1200  
                System call returned 0  
root@debian:~# rmmod captainHook  
root@debian:~#  
  
(root) 192.168.0.35 — Konsole <2>  
File Edit View Bookmarks Settings Help  
root@debian:~# dmesg -WH  
[Dec18 16:22] Captain Hook loaded successfully..  
[ +8.687024] Pname Syscall caught: HOOK!..HOOK!..HOOK!..HOOK!  
[ +4.819640] Pname Syscall caught: HOOK!..HOOK!..HOOK!..HOOK!  
[ +9.550080] Unloaded Captain Hook successfully
```

Như vậy chúng ta đã hook thành công!