بسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

آز سیستم‌های عامل

گزارش آزمایش هشتم بخش اول

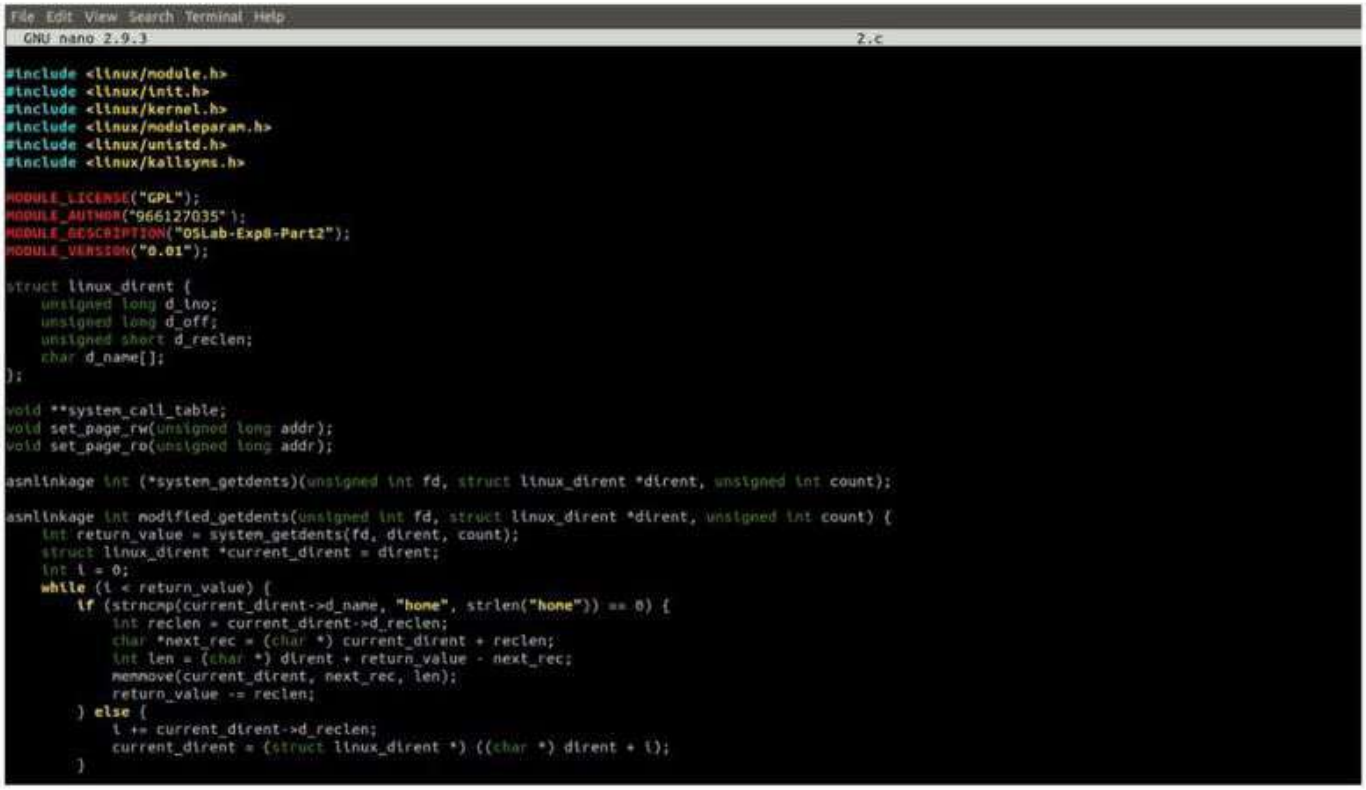
نگارش

امین منصوری

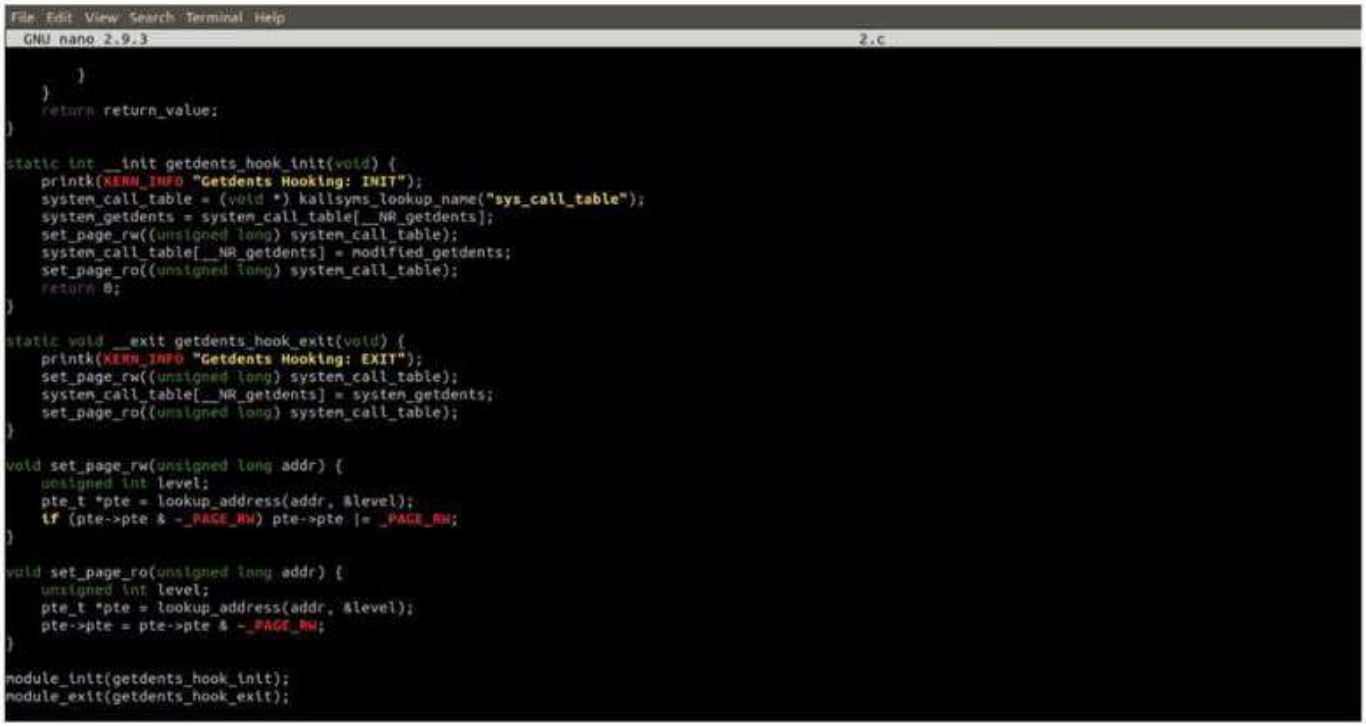
استاد

بیگی

کد ماژول مورد نظر را ایجاد می‌کنیم. این کد در تصویر زیر قابل مشاهده است:



```
File  Edit  View  Search  Terminal  Help
 GNU nano 2.9.3                                                    2.c

#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/moduleparam.h>
#include <linux/unistd.h>
#include <linux/kallsyms.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("966127035");
MODULE_DESCRIPTION("OSLab-Exp0-Part2");
MODULE_VERSION("0.01");

struct linux_dirent {
    unsigned long d_ino;
    unsigned long d_off;
    unsigned short d_reclen;
    char d_name[];
};

void **system_call_table;
void set_page_rw(unsigned long addr);
void set_page_ro(unsigned long addr);

asmlinkage int (*system_getdents)(unsigned int fd, struct linux_dirent *dirent, unsigned int count);

asmlinkage int modified_getdents(unsigned int fd, struct linux_dirent *dirent, unsigned int count) {
    int return_value = system_getdents(fd, dirent, count);
    struct linux_dirent *current_dirent = dirent;
    int i = 0;
    while (i < return_value) {
        if (strncmp(current_dirent->d_name, "hone", strlen("hone")) == 0) {
            int reclen = current_dirent->d_reclen;
            char *next_rec = (char *) current_dirent + reclen;
            int len = (char *) dirent + return_value - next_rec;
            memmove(current_dirent, next_rec, len);
            return_value -= reclen;
        } else {
            i += current_dirent->d_reclen;
            current_dirent = (struct linux_dirent *) ((char *) dirent + i);
        }
```



```
File  Edit  View  Search  Terminal  Help
 GNU nano 2.9.3                                                    2.c

        }
    }
    return return_value;
}

static int __init getdents_hook_init(void) {
    printk(KERN_INFO "Getdents Hooking: INIT");
    system_call_table = (void *) kallsyms_lookup_name("sys_call_table");
    system_getdents = system_call_table[__NR_getdents];
    set_page_rw((unsigned long) system_call_table);
    system_call_table[__NR_getdents] = modified_getdents;
    set_page_ro((unsigned long) system_call_table);
    return 0;
}

static void __exit getdents_hook_exit(void) {
    printk(KERN_INFO "Getdents Hooking: EXIT");
    set_page_rw((unsigned long) system_call_table);
    system_call_table[__NR_getdents] = system_getdents;
    set_page_ro((unsigned long) system_call_table);
}

void set_page_rw(unsigned long addr) {
    unsigned int level;
    pte_t *pte = lookup_address(addr, &level);
    if (pte->pte & ~_PAGE_RW) pte->pte |= _PAGE_RW;
}

void set_page_ro(unsigned long addr) {
    unsigned int level;
    pte_t *pte = lookup_address(addr, &level);
    pte->pte = pte->pte & ~_PAGE_RW;
}

module_init(getdents_hook_init);
module_exit(getdents_hook_exit);
```

در تصویر بالا، بعد از وارد کردن فایل‌های سرآیند، اطلاعات مربوط به ماژول مانند لایسنس، نام مولف، توضیحات را وارد می‌کنیم.

هدف اصلی این ماژول این است که getdents را قلاب کند. در متغیر system_getdents آدرس اصلی ماژول getdents را قرار داده‌ایم و با استفاده از متغیر modified_getdents و جایگزین کردن آن، عملکرد این تابع سیستمی را تغییر می‌دهیم.

درون حلقه‌ی موجود در modified_getdents تمامی تکرارهای کلمه home را بررسی کرده و از آن‌ها صرف نظر می‌کنیم. هم‌چنین، توجه کنید که برای آدرس‌های دیگر، از memmove استفاده کرده و آن‌ها را در خروجی نهایی لحاظ می‌کنیم. مقدار متغیر return_value نیز دائماً تنظیم می‌گردد تا دچار خطای حاصل از تغییر طول، به دلیل حذف یک مدخل نشویم.

در ادامه دو تابع getdents_hook_init و getdents_hook_exit را برای ماژول ایجاد می‌کنیم.

در تابع init، آدرس جدول فراخوانی‌های سیستمی را با کمک دستور kallsyms_lookup_name به دست می‌آوریم. سپس، آدرس اصلی مربوط به getdents را ذخیره کرده و در نهایت نسخه خاص خودمان را به جای آن می‌گذاریم. در تابع exit، عکس این فرایند انجام شده و همه چیز به حالت اولیه بر می‌گردد.

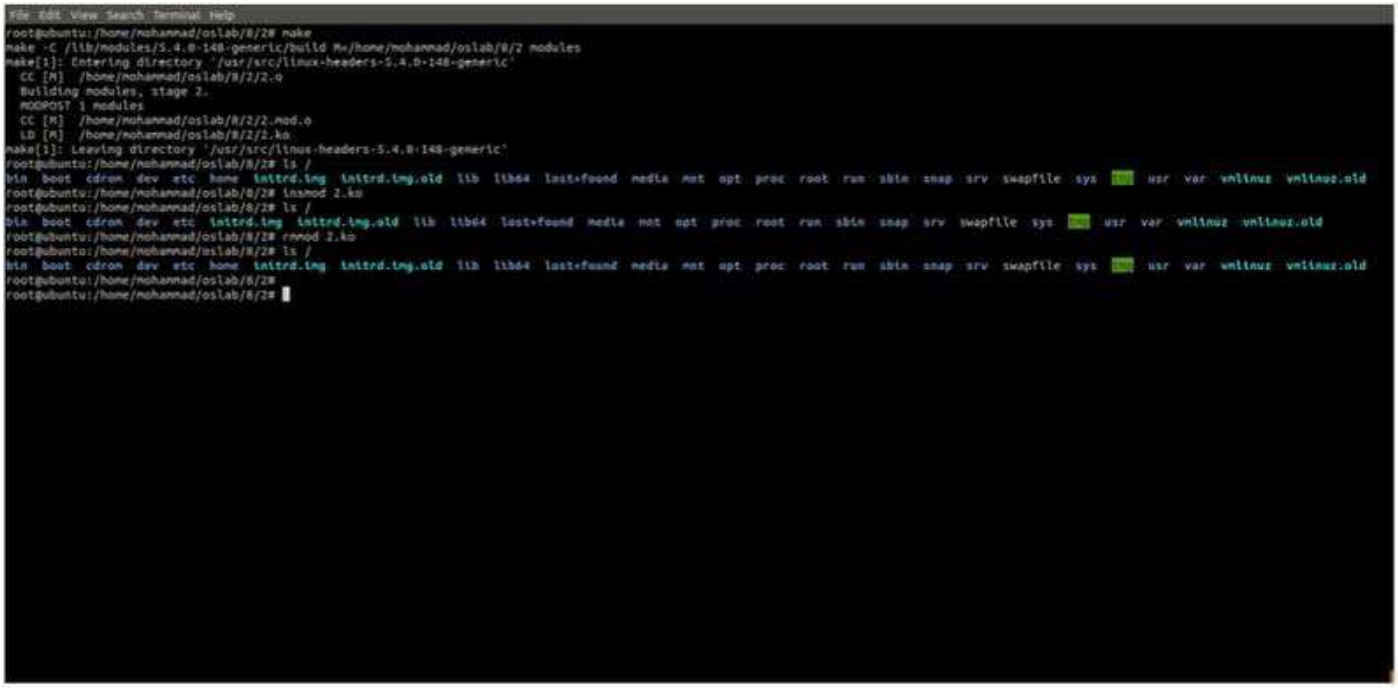
در نهایت، این دو تابع را در ورودی module_init و module_exit قرار می‌دهیم تا ماژول کامل شود.

بعد از ایجاد کد ماژول، نوبت به ایجاد کردن آن با دستور make می‌رسد. به همین منظور، ابتدا فایل Makefile را ایجاد می‌کنیم. تصویر کد مربوط به این فایل در زیر آمده است:

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3

CONFIG_MODULE_SIG=n

obj-m += 2.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
```

حال، دستور make را در ترمینال وارد می‌کنیم تا ماژول ساخته شود. سپس، قبل از لود کردن ماژول، یک بار دستور / ls را وارد می‌کنیم. در ادامه ماژول را با دستور insmod لود کرده و مجدداً دستور / ls را وارد می‌کنیم. انتظار داریم که دایکتوری home ناپدید شده باشد. سپس، ماژول را با دستور rmmod آن‌لود کرده و مجدداً دستور / ls را وارد می‌کنیم. انتظار داریم که دایکتوری home برگشته شده باشد.

در تصویر زیر می‌توانید عملکرد درست این ماژول را مورد بررسی قرار دهید.



سوال: ابزارهای ضد بدافزار و یا بدافزارها از این روش چه بهره‌ای می‌برند؟

تکنیک‌های مربوط به قلاب زدن (hooking) می‌توانند مهاجم را برای سرقت اطلاعات مهمی مانند رمز عبور، اطلاعات کارت بانکی و... کمک کنند. نحوه انجام این کار به این صورت است که ورودی‌های کاربر از کیبورد ثبت و ضبط می‌شوند. در ادامه، این شرایط فراهم می‌شود که پیش از ارسال بسته‌ها در شبکه، اطلاعات مهم داده‌ها به دست مهاجم برسد و بتواند از این داده‌ها در مقاصد شوم بهره ببرد.

در طرف دیگر، ابزارهای ضد بدافزار را داریم. این ابزارها در یک محیط sandbox در تلاش هستند تا با track کردن فراخوانی‌های سیستمی بدافزارها عملکرد آن‌ها را درک کنند. این کار با قرار دادن یک سری قلاب درایور در سطح کاربر یا هسته انجام می‌شود.