پروژه ی شماره ۲ درس آزمایشگاه سیستم عامل

شایان خورسندی ۹۴۵۲۳۱۲۶
میرسجاد ابوطالبی کهنه شهری ۹۴۵۲۳۰۰۹

گزارش:

در این پروژه با دستور cat به جهت چاپ محتوای داخل یک فایل و دو دستور echo و tee به جهت نوشتن مطالب در درون کار داریم. پس از اجرای هر کدام از دستورات زیر به کمک دستور strace می توانیم بفهمیم که چه فراخوانی های سیستمی در حین اجرای این دستورات فراخوانی میشوند. هدف ما این است که قبل از اجرای فراخوان های سیستمی مورد نظر خودمان تغییرات خودمان را به صورتی بدهیم که محتوای مورد نظر قبل از نوشته شدن بر روی فایل رمز گذاری و همچنین در حین خواندن قبل از نمایش اطلاعات به کاربر از حالت رمز گذاری شده خارج شود. بدین ترتیب کاربر متوجه رمز گذاری تا وقتی که ماژول بارگزاری شده است نخواهد شد. در حین اجرای هر کدام از دستورات زیر فراخوان های سیستمی زیر اجرا می شوند :

- فراخوان سیستمی sys_open جهت باز کردن یک فایل : در حین فراخوانی echo و tee این فراخوان سیستمی با مود ۰۶۶۶ فراخوانی میشود که می توان از این مود برای تشخیص فایل به جهت نوشته شدن یا خوانده شدن استفاده کرد. زیرا در حین cat که به جهت خواندن از روی فایل و نمایش آن برای کاربر استفاده میشود این مود استفاده نمی گردد.
- فراخوان سیستمی sys_read : این فراخوان سیستمی به جهت خواندن یک فایل اجرا میشود. Fd یک پارامتر این فراخوان مشخص کننده ی file descriptor به جهت std out یا std in یا هر چیز دیگر می باشند.
- فراخوان سیستمی sys_write : این فراخوان سیستمی به جهت نوشتن بر روی یک فایل اجرا می شود. Fd یک پارامتر این فراخوان مشخص کننده ی file descriptor به جهت std out یا std in یا هر چیز دیگر می باشد.

برای این که بتوانیم تشخیص دهیم که این فراخوان برای فایل مورد نظر ما استفاده میشود و تاثیری بر روی فایل ها ندارد، باید اول از همه مقصد خودمان را مشخص کنیم که پارامتر ماژول کرنل می باشد. همچنین برای این که متوجه شویم که فایل برای چه منظوری باز می شود از flag ۲ استفاده می کنیم تا متوجه شویم که فایل برای نوشتن و یا خواندن باز شده است. از این ۲ می توانیم تشخیص دهیم که در چه وضعیت هستیم و قرار است محتوای مورد نظر خودمان را رمز گذاری یا رمز گشایی کنیم. برای hook کردن فراخوانی های سیستمی مورد نظرمان از فایل syscall.s که در کرنل خود لینوکس وجود دارد به جهت استخراج signature فراخوان های سیستمی استفاده می کنیم.

```
static asmlinkage long (*real_sys_open)(const char __user *filename,
                        int flags, umode_t mode);
static asmlinkage long (*real_sys_read)(unsigned int fd, char __user *buf, size_t count);
static asmlinkage long (*real_sys_write)(unsigned int fd, const char __user *buf,
                        size_t count);
```

در بالا تصویر signature ۳ فراخوانی سیستمی مورد نظر را مشاهده می کنیم. تفاوت وجود داشته با signature اصلی این می باشد که به ابتدای نام تابع به دلخواه real اضافه کردیم و از نوع اشاره گر تعریف کردیم تا بعدا بتوانیم به آن دسترسی داشته باشیم. این تعریف حکم فراخوان سیستمی اصلی را دارد. همچنین یک تعریف مثل تعاریف بالا ایجاد می کنیم و به ابتدای نام آن ها fh اضافه می کنیم و در واقع همان فراخوان hook شده می باشد. تصویر fh_sys_open در زیر آمده است :

```
static asmlinkage long fh_sys_open(const char __user *filename, int flags, umode_t mode)
{
    long ret;
    int comp_result;
    comp_result = strcmp(filename, file_to_hook);
    if(comp_result == 0)
    {
        if(mode == 0666)
        {
            open_flag_for_write = 1;
            flag_for_read = 0;
        }
        else
        {
            open_flag_for_write = 0;
            flag_for_read = 1;
        }
        pr_info("File %s has been opened", file_to_hook);
        // pr_info("Value for open flag set to : %d", open_flag_for_write);
    }
    else
    {
        open_flag_for_write = 0;
        flag_for_read = 0;
    }
    ret = real_sys_open(filename, flags, mode);
    return ret;
}
```

در عکس بالا مشاهده می کنیم قبل از این فراخوان سیستمی اصلی به نام sys_open اجرا شود ما دستورات خودمان را اجرا می کنیم و سپس بعد از آن فراخوان سیستمی اصلی را اجرا می کنیم.

در هنگام اجرای دستور "echo "hello" | tee <filename>"، فراخوان سیستمی sys_write با مقدار fd 3 به جهت نوشتن مجتوا بر روی فایل مورد نظر اجرا می شود. بنابراین می توانیم قبل از اجرای این فراخوان سیستمی آن را hook کرده، سپس رمز گذاری کرده و در نهایت با اجرای فراخوان اصلی، محتوای رمز گذاری شده را بر روی فایل مورد نظر خودمان بنویسیم. کاربر از این اتفاق متوجه چیزی نمیشود به دلیل این که ما فراخوان sys_call که با fd مقدار ۱ اجرا میشود را hook نمی کنیم. همچنین با ۲ فلگ که در هنگام باز کردن فایل مقدار دهی می کنیم می توانیم بین sys_write در هنگام cat و echo-tee با یک fd تمایز قائل شویم.

همچنین در هنگام اجرای دستور "cat <filename>"، فراخوان سیستمی sys_write با مقدار fd 1 به جهت نوشتن محتوا برای کاربر به صورت standard output اجرا می شود. می توانیم قبل از اجرای این فراخوانی سیستمی آن را hook کنیم و محتوای مورد نظر را رمزگشایی کنیم و سپس فراخوان سیستمی اصلی را اجرا کنیم تا به کاربر محتوای اصلی نشان داده شود.

بنابراین اگر پیمانه ی کرنل فعال نباشد، در هنگام اجرای فراخوان سیستمی cat، رمز گشایی اجرا نمیشود و محتوای عجیب به کاربر نمایش داده میشود.

همچنین برای رمزنگاری و رمز گشایی از الگوریتم ساده ی ROT13 استفاده می شود که یک کاراکتر را با کاراکتر ۱۳ ام بعد از خودش تعویض می کند.