Hệ Điều Hành:

Đồ Án 2: Viết systemcall cho hệ thống và hook vào một system call có sẵn

Nguyễn Sĩ Hùng (1612226) - Đoàn Minh Hiếu (1612198)

Ngày 7 tháng 12 năm 2018

1 Syscall

Do mỗi lần cài đặt lại hai syscall pnametoid và pidtoname, ta cần phải build lại kernel và quá trình build rất tốn thời gian (1-2 tiếng) nên nhóm đã viết 2 syscalls pnametoid và pidtoname rồi thực hiện quá trình build một lần. Bên dưới là hai syscall cần phải cài đặt.

```
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/init.h>
#include <linux/string.h>
#include "pnametoid.h"
asmlinkage int sys_pnametoid(char* name){
    struct task_struct *task;
    for_each_process(task){
        if(strcmp(task->comm, name) == 0){
            return (int)task_pid_nr(task);
        }
    }
    return -1;
}
```

```
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <asm/uaccess.h>
#include <linux/init.h>
#include <linux/tty.h>
#include <linux/string.h>
#include <linux/pid.h>
#include <linux/pid_namespace.h>
#include "pidtoname.h"

asmlinkage int sys_pidtoname(int pid, char* buf, int len){
    struct task_struct *task;
    struct pid *pid_struct;
    int len_process_name=0;
    pid_struct = find_get_pid(pid);
    task = pid_task(pid_struct,PIDTYPE_PID);
    len_process_name = sprintf(buf,"%s",task->comm);
    if(len>len_process_name)
        return 0;
    if(len<=len_process_name)
        return len_process_name;
    return -1;
}
```

Sau khi đã viết xong 2 syscalls trên, ta tạo 2 thư mục có tên là pidtoname và pnametoid mới trong thư mục $/usr/src/linux_3$.16.36 (Tuỳ vào từng phiên bản kernel khác nhau sẽ có tên thư mục khác nhau). Sau đó, ta sẽ tạo file Makefile của từng syscall. Cấu trúc của file Makefile có dạng sau:

```
obj-y := {Tên file}.o
```

Tiếp đó, ta sẽ thêm 2 syscall này vào file $/usr/src/linux\_3.16.36/arch/x86/syscalls/syscall\_\times 64$ thông qua cấu trúc $< number >< abi >< name >< entrypoint >$. Đối với 2 syscall trên, ta sẽ thêm vào 2 dòng sau:

```
350     common  pnametoid               sys_pnametoid
351     common  pidtoname               sys_pidtoname
```

Tiếp theo, ta sẽ thêm vào 2 thư mục vào Makefile của kernel:

```
core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ pnametoid/ pidtoname/
```

Sau đó, ta sẽ thêm prototype của 2 syscalls này vào linux header. Tiếp đó, ta thực hiện quá trình build. Sau khi build thành công thì 2 syscall đã được cài vào hệ thống. Ta có thể test bằng cách tạo file sau:

```c
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <string.h>
int main(){
        printf("Choose the following options:\n1. pnametoid \n2. pidtoname\n");
        int choose;
        printf("You choose: ");
        scanf("%d", &choose);
        if(choose == 1){
                char name[100];
                printf("Enter your name: ");
                scanf("%s", name);
                strtok(name, "\n");
                int pid = syscall(350, name);
                printf("System call return %d\n", pid);
        }
        else if(choose == 2){
                char bufName[100];
                int pid;
                printf("Enter your pid: ");
                scanf("%d", &pid);
                int result_len = syscall(351, pid, bufName, 100);
                printf("System call returned %d\n", result_len);
                if(result_len > 0){
```

```
                printf("Your process name: %s\n", bufName);
            }
            else{
                printf("Cannot find the process with pid = %d\n", pid);
            }
        }
        return 0;
}
```

Kết quả khi test:

```
osboxes@osboxes:~$ ./test
Choose the following options:
1. pnametoid
2. pidtoname
You choose: 1
Enter your name: firefox
System call return 2969

osboxes@osboxes:~$ ./test
Choose the following options:
1. pnametoid
2. pidtoname
You choose: 2
Enter your pid: 2969
System call returned 7
Your process name: firefox
```

2 Hook vào một syscall có sẵn

Quá trình hook vào syscall open và write được thực hiện bằng cách tạo một module sẽ thay thế syscall của hệ thống bằng một syscall giả. Module này chứa các hàm có chức năng tìm syscall table của hệ thống, bật tắt khả năng ghi vào file syscall table này:

```
static void aquire_sys_call_table(void)
{
  unsigned long int offset;

  for (offset = PAGE_OFFSET; offset < ULLONG_MAX; offset += sizeof(void *))
    {
            sys_call_table = ( unsigned long ** ) offset;
            if (sys_call_table[ __NR_close ] == ( unsigned long * ) sys_close)
            break ;
    }
        printk(KERN_EMERG "Syscall Table Address: %p\n", sys_call_table);
}
```

```
static void allow_writing( void )
{
  write_cr0( read_cr0() & ~0x10000 );
}

static void disallow_writing( void )
{
  write_cr0( read_cr0() | 0x10000 );
}
```

Sau đó, ta sẽ viết lại các syscall open và write:

```
asmlinkage int        new_write(unsigned int fd, const char __user *buf, size_t nBytes)
{
printk(KERN_INFO "PROCESS NAME: %s\n, FILE DISCRIPTION: %d\n, nBYTES: %d\n\n", current->comm
return original_write(fd, buf, nBytes);
}

asmlinkage int new_open(const char __user * filename, int flags, mode_t mode)
{
printk( KERN_INFO "PROCESS NAME: %s FILE OPEN: %s\n", current->comm, filename);
return original_open(filename, flags, mode);
}
```

Cuối cùng ta sẽ thêm các hàm init và exit:

```
static int init_mod( void )
{
  aquire_sys_call_table();

  original_write = ( void * ) sys_call_table[ __NR_write ];
  original_open = ( void * ) sys_call_table[ __NR_open ];

  allow_writing();

  sys_call_table[ __NR_write ] = ( unsigned long * ) new_write;
  sys_call_table[ __NR_open ] = ( unsigned long * ) new_open;

  disallow_writing();

  return 0;
}

static void exit_mod( void )
{
  allow_writing();
```

```
  sys_call_table[ __NR_write ] = ( unsigned long * ) original_write;
  sys_call_table[ __NR_open ] = ( unsigned long * ) original_open;

  disallow_writing();
}
```

Từ đó, ta tạo file Makefile:

```
\begin{minted}{C}
obj-m += rootkit.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
```

Sau đó, ta sẽ tạo file test các hàm syscall open và write của hệ thống. File này sẽ gọi các hàm open và write:

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <string.h>
int main(){
int fd = open("in.txt",O_WRONLY | O_CREAT | O_APPEND);
printf("FILE DESCRIPTION: %d\n",fd);
if(write(fd, "He Dieu Hanh\n", 13) == 13) {
    printf("WRITE SUCCESSFULLY\n");
}
else{
    printf("WRITE FAILURE\n");
}
return 0;
}
```

Ta sẽ test thử file test có chạy thành công hay chưa?

```
FILE DESCRIPTION: 3
WRITE SUCCESSFULLY
```

Cuối cùng, ta sẽ kiểm tra dmesg:

```
osboxes@osboxes:~$ dmesg | grep gnome-terminal
[ 3939.537035] PROCESS NAME: gnome-terminal FILE: 3 nBYTES: 13
osboxes@osboxes:~$ dmesg | grep 'test'
[ 4295.236764] PROCESS NAME: test FILE OPEN: in.txt
```