添加 Linux 系统调用（X86）——实现系统调用日志收集

作者：zhang-ke-wei
原文链接：https://ld246.com/article/1596538269153
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

第一步 打开系统调用表表项文件增加系统调用表表项

sudo vim arch/x86/entry/syscalls/syscall_64.tbl

添加335号系统调用：

```
335     common  myaudit                 __x64_sys_myaudit
```

第二步 添加系统调用函数

sudo vim arch/x86/kernel/myaudit.c

myaudit.c :

```
#include <linux/uaccess.h>
#include <linux/proc_fs.h>
#include <linux/init.h>
#include <linux/types.h>
#include <linux/sched.h>
#include <linux/syscalls.h>
#include <linux/kernel.h>
#include <asm/current.h>

void (*my_audit)(int,int) = 0;

int(*my_sysaudit)(u8,u8 *,u16,u8) = 0;
SYSCALL_DEFINE4(myaudit,u8,type,u8 *,us_buf,u16,us_buf_size,u8,reset)
{
        if (my_audit){
                printk("IN KENEL:my system call sys_myaudit() working\n");
                return (*my_sysaudit)(type,us_buf,us_buf_size,reset);

        } else
                printk("my_audit is not exist\n");
        return 1;

}

EXPORT_SYMBOL(my_audit);
EXPORT_SYMBOL(my_sysaudit);
```

第三步 修改Makefile文件

sudo vim arch/x86/kernel/Makefile

```
obj-y                   += myaudit.o
```

第四步 增加函数声明

sudo vim include/linux/syscalls.h

添加文件最后添加以下函数声明（#endif前）

```
asmlinkage long sys_myaudit(u8,u8 *,u16,u8);
extern void (*my_audit)(int,int);
```

第五步 拦截相关系统调用

```
sudo vim arch/x86/entry/common.c
```

在do_syscall_64函数中添加代码：

```
if (likely(nr < NR_syscalls)) {
		nr = array_index_nospec(nr, NR_syscalls);
		regs->ax = sys_call_table[nr](regs);

		//add myself code
		if( nr==2 || nr==3 || nr==39 || nr==56 || nr== 57 || nr==59){
			if(my_audit){
				(*my_audit)(nr,regs->ax);
			}else{
				printk("my_audit is not exist");
			}
		}
#ifdef CONFIG_X86_X32_ABI
	}
```

第六步 编译内核

```
make
```

第七步 安装内核

```
sudo make modules_install
sudo make install
```

第八步 重启系统

```
sudo reboot
```

第九步 编写内核模块实现系统调用

my_audit.c：

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/sched.h>
#include <linux/uaccess.h>

#define COMM_SIZE 16
#define AUDIT_BUF_SIZE 20
```

```
MODULE_LICENSE("GPL v2");

struct syscall_buf{
    u32 serial;
    u64 ts_sec;
    u64 ts_micro;
    u32 syscall;
    u32 status;
    pid_t pid;
    uid_t uid;
    u8 comm[COMM_SIZE];
};

DECLARE_WAIT_QUEUE_HEAD(buffer_wait);

static struct  syscall_buf  audit_buf[AUDIT_BUF_SIZE];
static int current_pos = 0;
static u32 serial = 0;

void syscall_audit(int syscall,int return_status)
{
    struct syscall_buf *ppb_temp;
    struct timespec64 nowtime;

    ktime_get_real_ts64(&nowtime);
    if(current_pos<AUDIT_BUF_SIZE){
        ppb_temp = &audit_buf[current_pos];
        ppb_temp->serial = serial++;
        ppb_temp->ts_sec = nowtime.tv_sec;
        ppb_temp->ts_micro = nowtime.tv_nsec;
        ppb_temp->syscall = syscall;
        ppb_temp->status = return_status;
        ppb_temp->pid = current->pid;
        ppb_temp->uid = current->tgid;

        memcpy(ppb_temp->comm,current->comm,COMM_SIZE);
        if(++current_pos ==AUDIT_BUF_SIZE * 6/10)
            {
                printk("IN MODULE_AUDIT:yes,it near full\n");
                wake_up_interruptible(&buffer_wait);
            }
        }

}

int sys_audit(u8 type,u8 *us_buf,u16 us_buf_size,u8 reset)
{
    int ret = 0;
        if(!type){
            if(clear_user(us_buf,us_buf_size)){
                printk("Error:clear_user\n");
                return 0;
            }
```

```
            printk("IN MODULE_systemcall:starting...\n");
            ret = wait_event_interruptible(buffer_wait,current_pos >= AUDIT_BUF_SIZE *6/10);
            printk("IN MODULE_systemcall:over,current_pos is %d\n",current_pos);
            if(copy_to_user(us_buf,audit_buf,(current_pos)*sizeof(struct syscall_buf))){
                printk("Error:copy error\n");
                return 0;
            }
            ret = current_pos;
            current_pos = 0;

        }
        return ret;
}

extern void(*my_audit)(int,int);
extern int (*my_sysaudit)(u8,u8*,u16,u8);
static int __init audit_init(void)
{
    my_sysaudit = sys_audit;
    my_audit = syscall_audit;
    printk("Exiting System Call Auditing\n");
    return;
}

module_init(audit_init);

static void __exit audit_exit(void)
{
    my_audit = NULL;
    my_sysaudit = NULL;
    printk("Exiting System Calling Auditing\n");
    return;
}

module_exit(audit_exit);
```

Makefile：

```
obj-m:= my_audit.o
CURRENT_PATH:=$(shell pwd)
LINUX_KERNEL_PATH:=/opt/linux-5.4

all:
        make -C $(LINUX_KERNEL_PATH) M=$(CURRENT_PATH) modules
clean:
        make -C $(LINUX_KERNEL_PATH) M=$(CURRENT_PATH) clean
```

第十步 编译安装内核模块

```
make
insmod my_audit.ko
```

第十一步 编写用户空间代码测试系统调用

```c
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <signal.h>
#include <time.h>
#include <sys/resource.h>
#include <sys/syscall.h>
#include <sys/types.h>

#define COMM_SIZE 16

typedef unsigned char u8;
typedef unsigned int u32;
typedef unsigned long long u64;

struct syscall_buf{
    u32 serial;
    u64 ts_sec;
    u64 ts_micro;
    u32 syscall;
    u32 status;
    pid_t pid;
    uid_t uid;
    u8 comm[COMM_SIZE];
};

#define AUDIT_BUF_SIZE (20*sizeof(struct syscall_buf))

int main(int argc,char *argv[])
{
    u8 col_buf[AUDIT_BUF_SIZE];
    unsigned char reset = 1;
    int num = 0;
    int i,j;
    struct syscall_buf *p;
    while(1){
        num = syscall(335,0,col_buf,AUDIT_BUF_SIZE,reset);
        printf("num :%d\n",num);
        p = (struct syscall_buf *)col_buf;
        for(i=0;i<num;i++){
            printf("num[%d],serial:%d,\t syscall :%d,\t pid:%d,\t comm:%s,\t time:%s\n",i,p[i].serial,
[i].syscall,p[i].pid,p[i].comm,ctime(&p[i].ts_sec));
        }
    }
    return 1;
}
```

结果：

```
root@hecs-x-medium-2-linux-20200628151506:/home/zkw/workspace/syscall# make
make -C /opt/linux-5.4 M=/home/zkw/workspace/syscall modules
make[1]: Entering directory '/opt/linux-5.4'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/opt/linux-5.4'
root@hecs-x-medium-2-linux-20200628151506:/home/zkw/workspace/syscall# ls
Makefile       Module.symvers  myaudit.c    my_audit.mod    my_audit.mod.o  test_syscall
modules.order  my_audit.c      my_audit.ko  my_audit.mod.c  my_audit.o      test_syscall.c
root@hecs-x-medium-2-linux-20200628151506:/home/zkw/workspace/syscall# insmod my_audit.ko
```

```
num[11],serial:474,     syscall :3,     pid:25111,     comm:df,       time:Tue Aug  4 18:42:41 2020

num :12
num[0],serial:475,      syscall :56,    pid:16706,     comm:bash,     time:Tue Aug  4 18:42:41 2020

num[1],serial:476,      syscall :59,    pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[2],serial:477,      syscall :2,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[3],serial:478,      syscall :3,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[4],serial:479,      syscall :2,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[5],serial:480,      syscall :3,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[6],serial:481,      syscall :2,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[7],serial:482,      syscall :3,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[8],serial:483,      syscall :2,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[9],serial:484,      syscall :3,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[10],serial:485,     syscall :2,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020

num[11],serial:486,     syscall :3,     pid:25112,     comm:who,      time:Tue Aug  4 18:42:41 2020
```

参考

陈莉君教授的Linux内核分析与应用第六章 系统调用~~~~