Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 4. 19

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

< 모니터 해킹하기!>

monitor_hack.c

```
// 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>

#include <asm/current.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 *))
	{
		p = (unsigned long *)tmp;
		if(p[__NR_close] == (unsigned long)sys_close)
		{
			sys_table = (unsigned long **)p;
			return &sys_table[0];
		}
	}

	return NULL;
}

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);
	printk(KERN_DEBUG "current task = 0x%x\n", current);
#if 0
	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);
#endif

        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)
        {
                printk("<0>Can't find sys_call_table\n");
                return -1;
        }

        printk("<0>sys_call_table is at[%p]\n", sys_call_table);

        // CR0 레지스터를 읽어옴
        cr0 = read_cr0();
        // Page 쓰기를 허용함
        write_cr0(cr0 & ~0x00010000);

        /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 설정함 */
        fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
        if(!fixed_set_memory_rw)
        {
                printk("<0>Unable to find set_memory_rw symbol\n");
                return 0;
        }

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

        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");
```

- c 파일을 이렇게 만들어 준다.
- Makefile 을 만들어 준다.
- make 명령어를 친다.
- ko 파일(kernel object) 파일을 insmod 한다.

(하는 순간 모니터가 해킹이 된다.)

< 커널 드라이빙 하기 >

```c
int syscall_hooking_init(void)
{
    unsigned long cr0;

    if((sys_call_table = locate_sys_call_table()) == NULL)
    {
        printk("<0>Can't find sys_call_table\n");
        return -1;
    }

    printk("<0>sys_call_table is at[%p]\n", sys_call_table);

    // CR0 레지스터를 읽어옴
    cr0 = read_cr0();
    // Page 쓰기를 허용함
    write_cr0(cr0 & ~0x00010000);

    /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 설정함 */
    fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
    if(!fixed_set_memory_rw)
    {
        printk("<0>Unable to find set_memory_rw symbol\n");
        return 0;
    }

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

    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;
}
```

syscall_hooking_init(void) 함수를 드라이빙 한다.

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

일단 read_cr0()부터 본다.

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

native_read_cr0()를 본다.

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

=r 은 레지스터 , =m 는 메모리이다.

```
  // Page 쓰기를 허용함
  write_cr0(cr0 & ~0x00010000);
```

write_cr0 를 찾아 본다. 인자는 레지스터를 정렬한 값을 넣는다.

```
static inline void write_cr0(unsigned long x)
{
    native_write_cr0(x);
}
```

그대로 들어간다.

```
static inline void native_write_cr0(unsigned long val)
{
    asm volatile("mov %0,%%cr0": : "r" (val), "m" (__force_order));
}
```

안에 보니 어셈블리 명령어로 이루어져 있다.

```
  /* set_memory_rw 라는 심볼을 찾아와서 fixed_set_memory_rw 에 설정함 */
  fixed_set_memory_rw = (void *)kallsyms_lookup_name("set_memory_rw");
```

그다음 순서인 kallsyms_lookup_name 을 보기 전 인자를 본다.. (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);
}
```