Linux Rootkit

Adrien '*schischi*' Schildknecht

July 17, 2015

Section 1

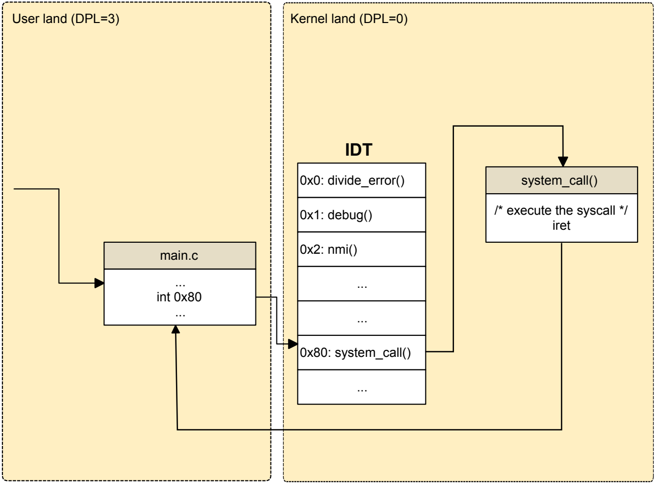
IDT hooking

Why?

Main interface between the kernel and the world (userland, hardware. . .)

Modifying the IDT

- The Address of the IDT is stored in a register;
- Changing an entries:
  - Modify the table (RO);
  - Create a new table;



User land (DPL=3)

Kernel land (DPL=0)

main.c

...
int 0x80
...

IDT

0x0: divide_error()

0x1: debug()

0x2: nmi()

...

...

0x80: system_call()

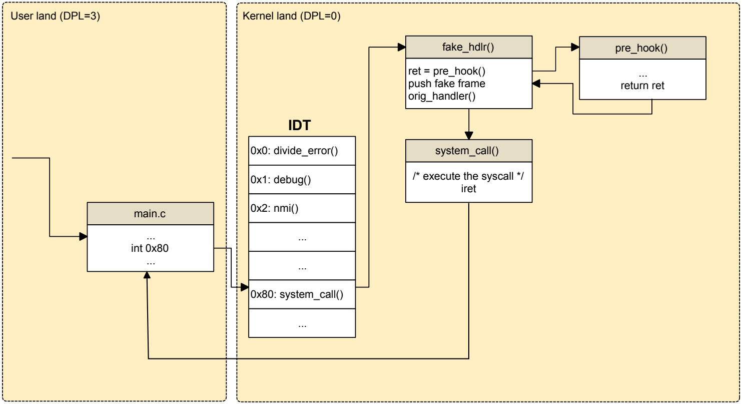
...

system_call()

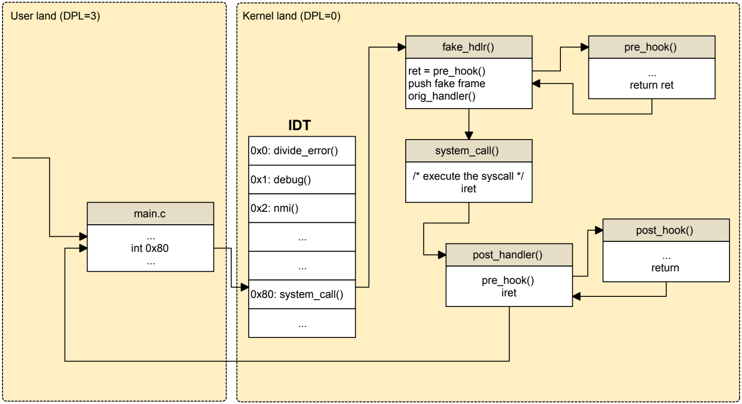
/* execute the syscall */
iret

Interrupt stack frame

Pre hook


User land (DPL=3)
Kernel land (DPL=0)
main.c
...
int 0x80
...
IDT
0x0: divide_error()
0x1: debug()
0x2: nmi()
...
...
0x80: system_call()
...
fake_hdlr()
ret = pre_hook()
push fake frame
orig_handler()
pre_hook()
...
return ret
system_call()
/* execute the syscall */
iret

Post hook


User land (DPL=3)
Kernel land (DPL=0)
main.c
...
int 0x80
...
IDT
0x0: divide_error()
0x1: debug()
0x2: nmi()
...
...
0x80: system_call()
...
fake_hdlr()
ret = pre_hook()
push fake frame
orig_handler()
pre_hook()
...
return ret
system_call()
/* execute the syscall */
iret
post_handler()
pre_hook()
iret
post_hook()
...
return

Summary

Section 2

Syscall hooking

How to make a syscall

3 ways:

- 32bits: int 0x80, sysenter (Intel), syscall (AMD);
- 64bits: syscall;

Int 0x80

Sysenter

```
    /* Obtain a valid pointer to per cpu data*/
    swapgs
    /* Setup a stack */
    mov $stack_sysenter, %rsp
    add %gs:this_cpu_off, %rsp
    /* Save registers on the stack */
    sub $0x28, %rsp  /* Skip exception frame */
    SAVE_REGS
    /* Fill exception frame */
    movl 12(%rbp),  %eax        /* RIP */
    movq %rax,      0x80(%rsp)
    movq $0x23,     0x88(%rsp)  /* CS */
    movq $0x0,      0x90(%rsp)  /* RFLAGS */
    movl 0x0(%rbp), %eax        /* RSP */
    movq %rax,      0x98(%rsp)
    movq $0x2b,     0xa0(%rsp)  /* SS */
    mov %rsp, %rdi
    /* Set an invalid esp as return addr */
    movl $__stringify(0x42cafe42), 12(%rbp)
    /* Pre-hook ! */
    call *sysenter_pre_hook
    RESTORE_REGS
    /* Call the original handler without swapgs */
    jmp *(sysenter_orig_hdlr + 3)

```

Syscall

Section 3

Conclusion

Conclusion

```c
#define MEGA(S) ((S) * 1024 * 1024)

int main(int argc, char *argv[]) {
    char buf[4096];
    int fd = open("/home/schischi/foo", O_CREAT | O_WRONLY, 0660);

    if (argc == 2 && !strcmp(argv[1], "-f"))
        if (fallocate(fd, 0, 0, MEGA(700)) != 0)
            return 1;
    for (int i = 0; i < MEGA(700) / sizeof (buf); ++i)
        write(fd, buf, 4096);
    write(fd, buf, MEGA(700) % sizeof (buf));

    unlink("/home/schischi/foo");
    return 0;
}

```

```
$ repeat 100; ./a.out
    ./a.out  0.01s user 1.46s system 18% cpu 8.018 total

$ repeat 100; ./a.out -f
```

Conclusion

Questions ?

schischi@lse.epita.fr

schischi - irc.rezosup.org

References

- FS design
  - Book "Practical File System Design" by Dominic Giampaolo
- VFS
  - http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git
  - http://lwn.net/Kernel/Index/
- Journaling, logging
  - http://pages.cs.wisc.edu/~remzi/OSTEP/file-lfs.pdf
  - http://research.cs.wisc.edu/wind/Publications/sba-usenix05.pdf
- Ext4
  - https://ext4.wiki.kernel.org/index.php/Ext4_Design
  - http://www.ibm.com/developerworks/library/l-anatomy-ext4/
- Btrfs
  - http://video.linux.com/videos/chris-mason-btrfs-file-system
  - http://atrey.karlin.mff.cuni.cz/~jack/papers/lk2009-ext4-btrfs.pdf