Android smart platform kernel malware

Android platform based linux kernel rootkit

2011. 10.

Dong-Hoon You*, Bong-Nam Noh**
*INetCop Security
**SSRC, Chonnam National University

Contents

- Android smart platform kernel malware
  - Introduction
  - Technical Description
  - Demonstration
  - Conclusion

Introduction

Android smart platform sandbox

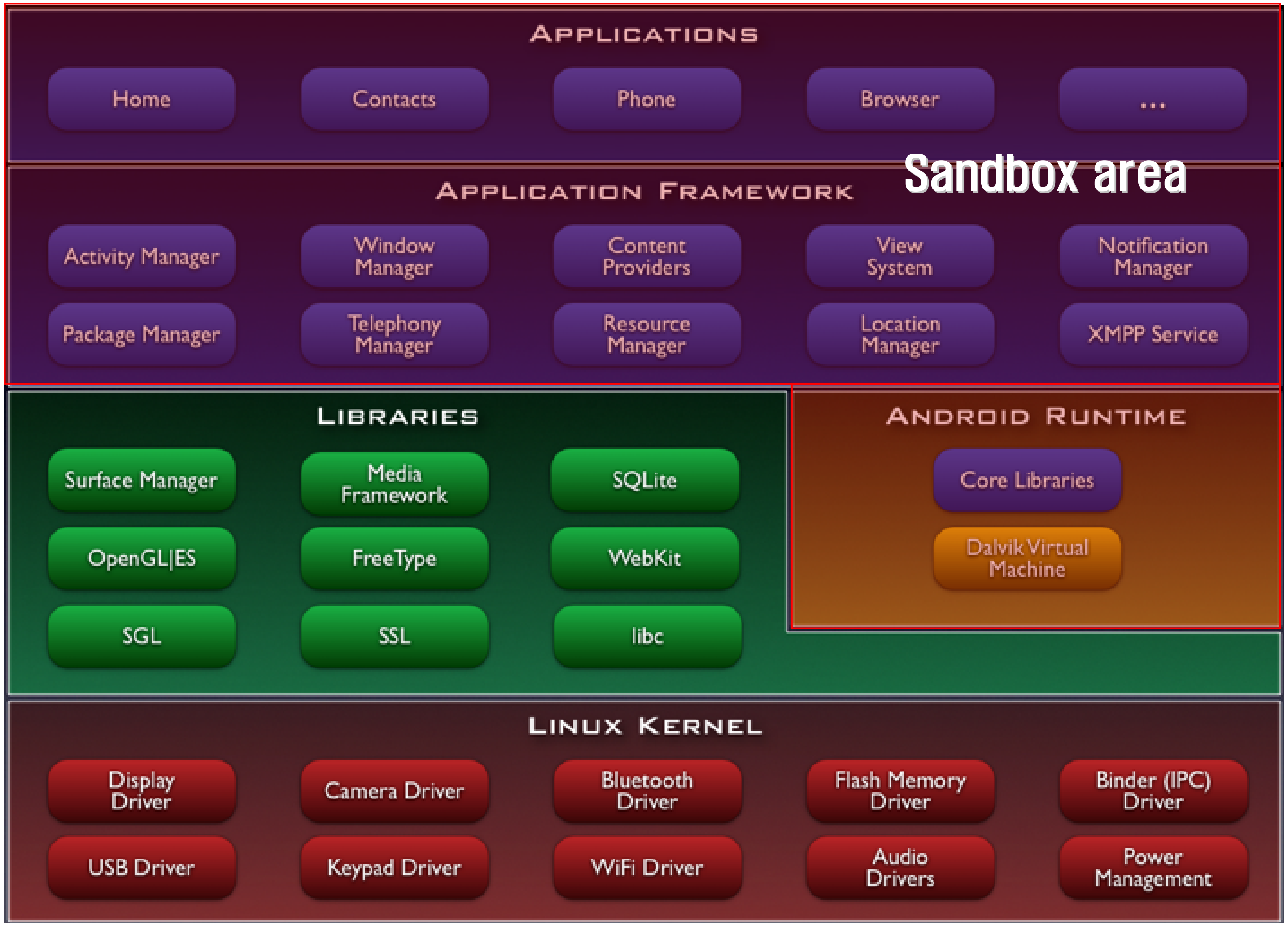
Android sandbox 1/4

The purpose of Sandbox

The ideal operating of a sandbox

Android sandbox 2/4

- Android smart platform sandbox



Applications

Home | Contacts | Phone | Browser | ...

Sandbox area

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource Manager | Location Manager | XMPP Service

Libraries

Surface Manager | Media Framework | SQLite

OpenGL|ES | FreeType | WebKit

SGL | SSL | libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver | Camera Driver | Bluetooth Driver | Flash Memory Driver | Binder (IPC) Driver

USB Driver | Keypad Driver | WiFi Driver | Audio Drivers | Power Management

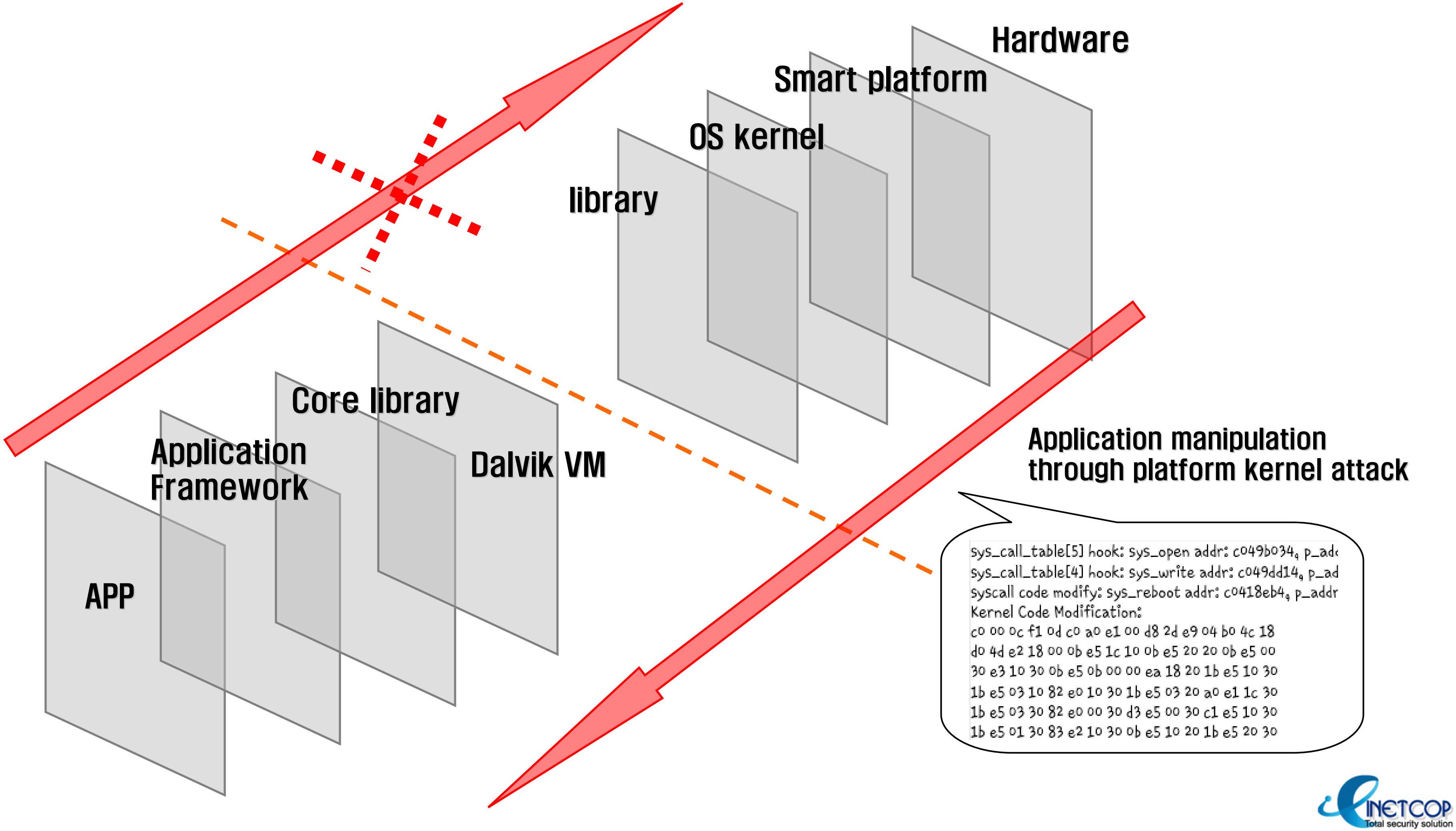
Android sandbox 3/4

Reality of sandbox security

However, the reality is a battlefield

Android sandbox 4/4

- **Security Technology Issue**



Application manipulation through platform kernel attack

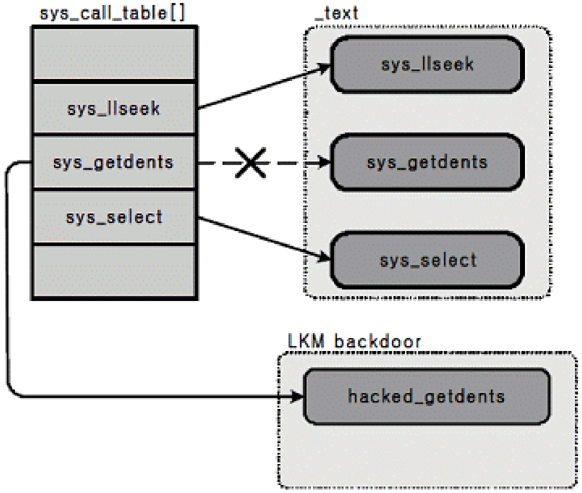
```
sys_call_table[5] hook: sys_open addr: c049b034, p_ad
sys_call_table[4] hook: sys_write addr: c049dd14, p_ad
syscall code modify: sys_reboot addr: c0418eb4, p_addr
Kernel Code Modification:
c0 00 0c f1 0d c0 a0 e1 00 d8 2d e9 04 b0 4c 18
d0 4d e2 18 00 0b e5 1c 10 0b e5 20 20 0b e5 00
30 e3 10 30 0b e5 0b 00 00 ea 18 20 1b e5 10 30
1b e5 03 10 82 e0 10 30 1b e5 03 20 a0 e1 1c 30
1b e5 03 30 82 e0 00 30 d3 e5 00 30 c1 e5 10 30
1b e5 01 30 83 e2 10 30 0b e5 10 20 1b e5 20 30
```

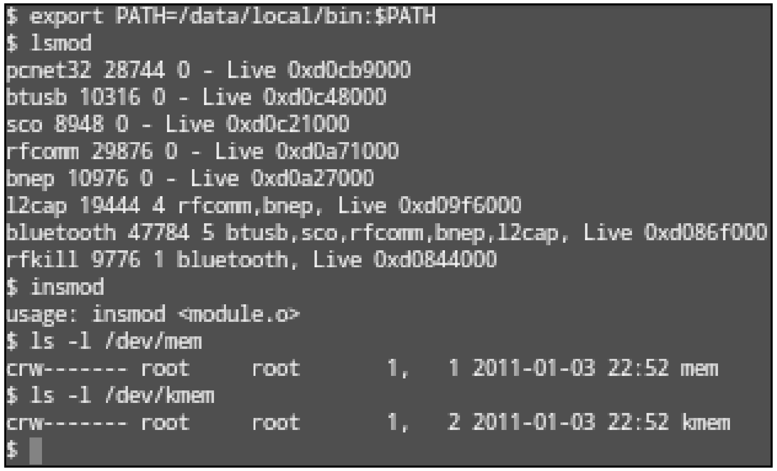
Technical Description

Kernel Level Hooking Techniques

Android Kernel Rootkit 1/2

- **LKM (Loadable kernel module) access technic**
  - Can both add or remove a new code without a kernel compile or reboot
  - Change function addr in sys_call_table into hacker's function and hook it
- **KMEM device access technic**
  - Silvio Cesare's "RUNTIME KMEM PATCHING" / sd's Phrack 58-7
  - Access via /dev/mem(physical), /dev/kmem(virtual) memory mapping files

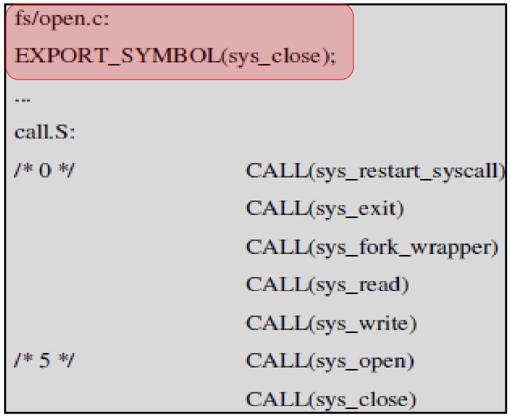




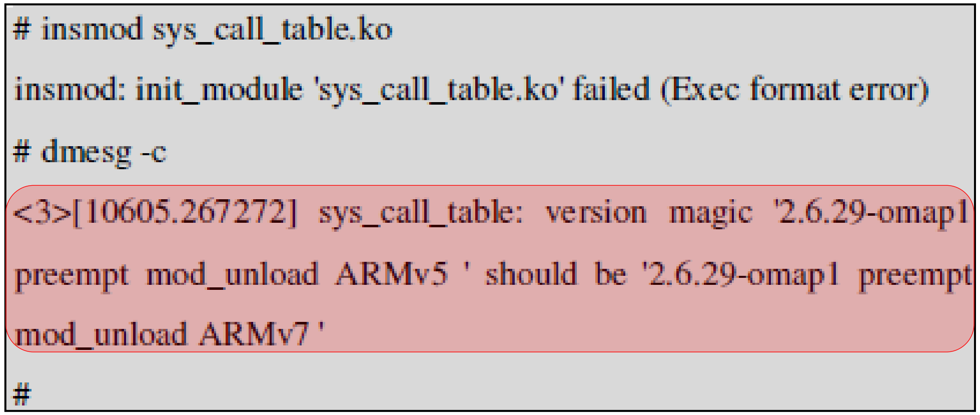
```
$ export PATH=/data/local/bin:$PATH
$ lsmod
pcnet32 28744 0 - Live 0xd0cb9000
btusb 10316 0 - Live 0xd0c48000
sco 8948 0 - Live 0xd0c21000
rfcomm 29876 0 - Live 0xd0a71000
bnep 10976 0 - Live 0xd0a27000
l2cap 19444 4 rfcomm,bnep, Live 0xd09f6000
bluetooth 47784 5 btusb,sco,rfcomm,bnep,l2cap, Live 0xd086f000
rfkill 9776 1 bluetooth, Live 0xd0844000
$ insmod
usage: insmod <module.o>
$ ls -l /dev/mem
crw------- root     root       1,   1 2011-01-03 22:52 mem
$ ls -l /dev/kmem
crw------- root     root       1,   2 2011-01-03 22:52 kmem
$ 
```

Android Kernel Rootkit 2/2

- **Searching sys_call_table**
  - Getting sys_call_table address in vector_swi handler
  - Finding sys_call_table address through sys_close address searching
- **Treating version magic**
  - Modification of UTS_RELEASE value in utsrelease.h header
  - Modification of __module_depends value in the kernel module
  - Direct overwrite of vermagic value in a compiled kernel modile binary



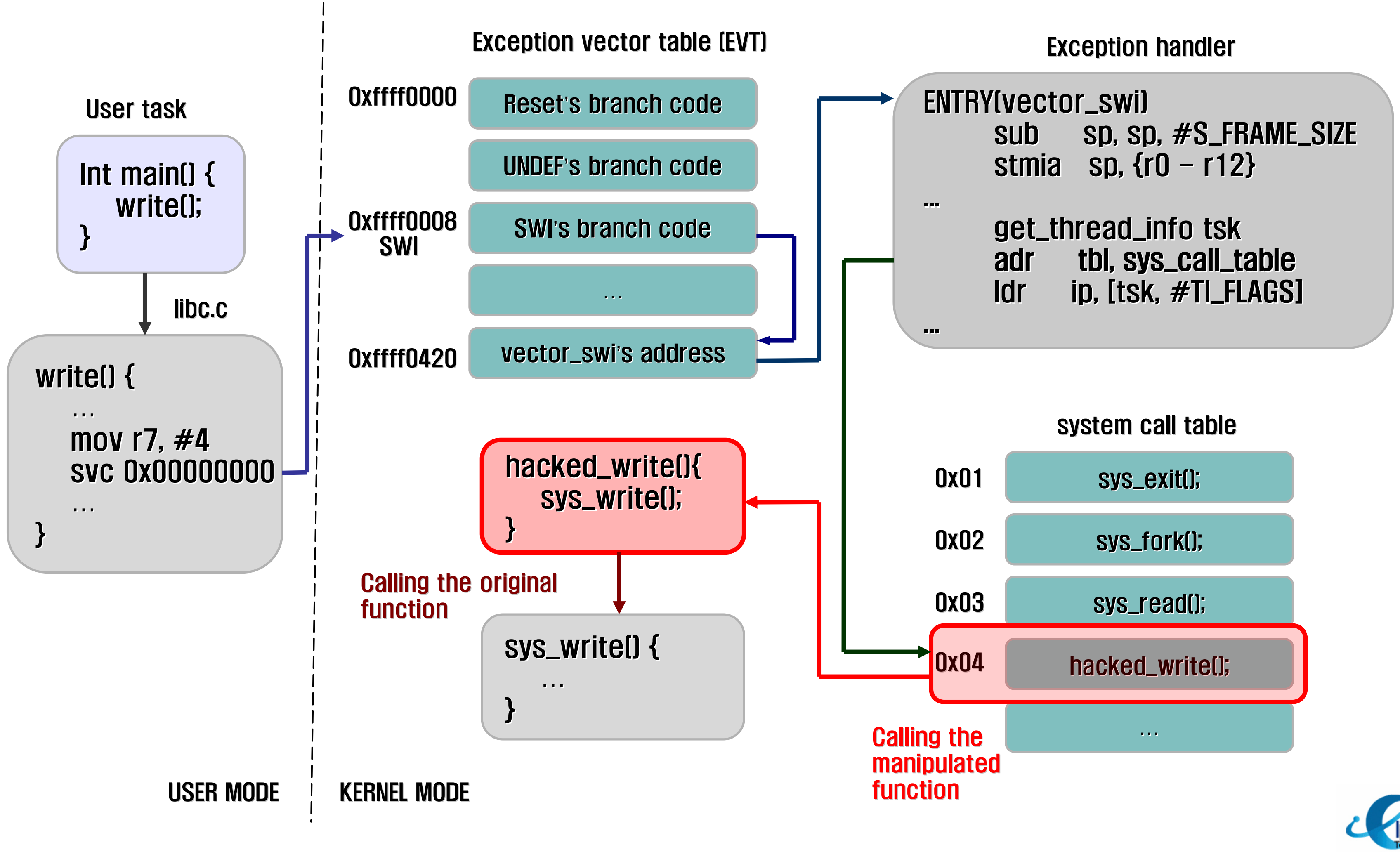
```
fs/open.c:
EXPORT_SYMBOL(sys_close);
...
call.S:
/* 0 */            CALL(sys_restart_syscall)
                   CALL(sys_exit)
                   CALL(sys_fork_wrapper)
                   CALL(sys_read)
                   CALL(sys_write)
/* 5 */            CALL(sys_open)
                   CALL(sys_close)
```



```
# insmod sys_call_table.ko
insmod: init_module 'sys_call_table.ko' failed (Exec format error)
# dmesg -c
<3>[10605.267272] sys_call_table: version magic '2.6.29-omap1
preempt mod_unload ARMv5 ' should be '2.6.29-omap1 preempt
mod_unload ARMv7 '
#
```

Hooking Techniques 1/4

- Basic techniques for sys_call_table hooking



User task

```
Int main() {
     write();
}
```

libc.c

```
write() {
     …
     mov r7, #4
     svc 0x00000000
     …
}
```

Exception vector table (EVT)

0xffff0000	Reset's branch code
	UNDEF's branch code
0xffff0008 SWI	SWI's branch code
	…
0xffff0420	vector_swi's address

Exception handler

```
ENTRY(vector_swi)
          sub       sp, sp, #S_FRAME_SIZE
          stmia     sp, {r0 – r12}
…
          get_thread_info tsk
          adr       tbl, sys_call_table
          ldr       ip, [tsk, #TI_FLAGS]
…
```

```
hacked_write(){
     sys_write();
}
```

Calling the original function

```
sys_write() {
     …
}
```

system call table

0x01	sys_exit();
0x02	sys_fork();
0x03	sys_read();
0x04	hacked_write();
	…

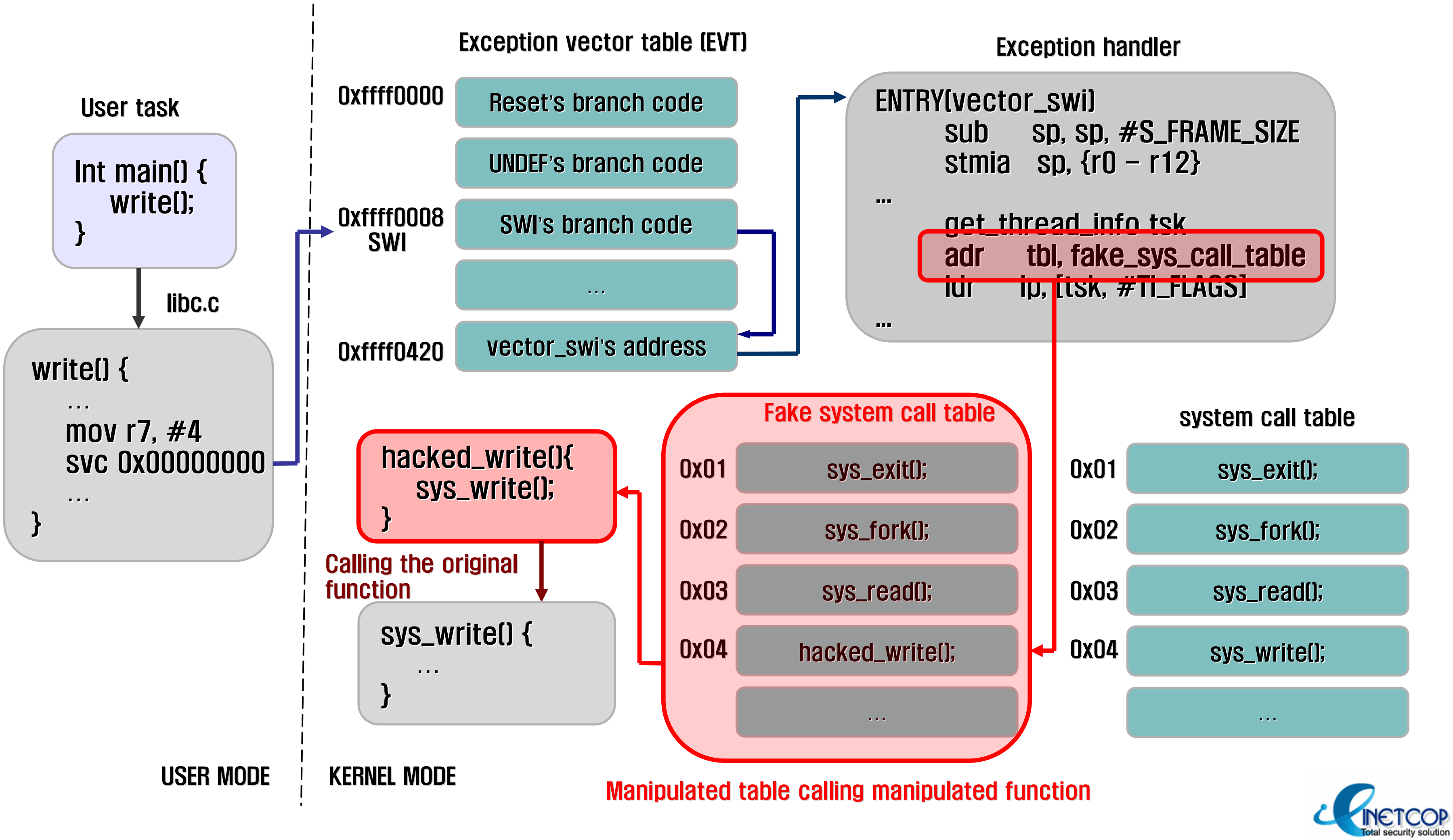
Calling the manipulated function

USER MODE

KERNEL MODE

Hooking Techniques 2/4

- **Modifying vector_swi handler routine**



```
Int main() {
    write();
}
```

```
write() {
    …
    mov r7, #4
    svc 0x00000000
    …
}
```

```
ENTRY(vector_swi)
        sub     sp, sp, #S_FRAME_SIZE
        stmia   sp, {r0 – r12}
…
        get_thread_info tsk
        adr     tbl, fake_sys_call_table
        ldr     ip, [tsk, #TI_FLAGS]
…
```

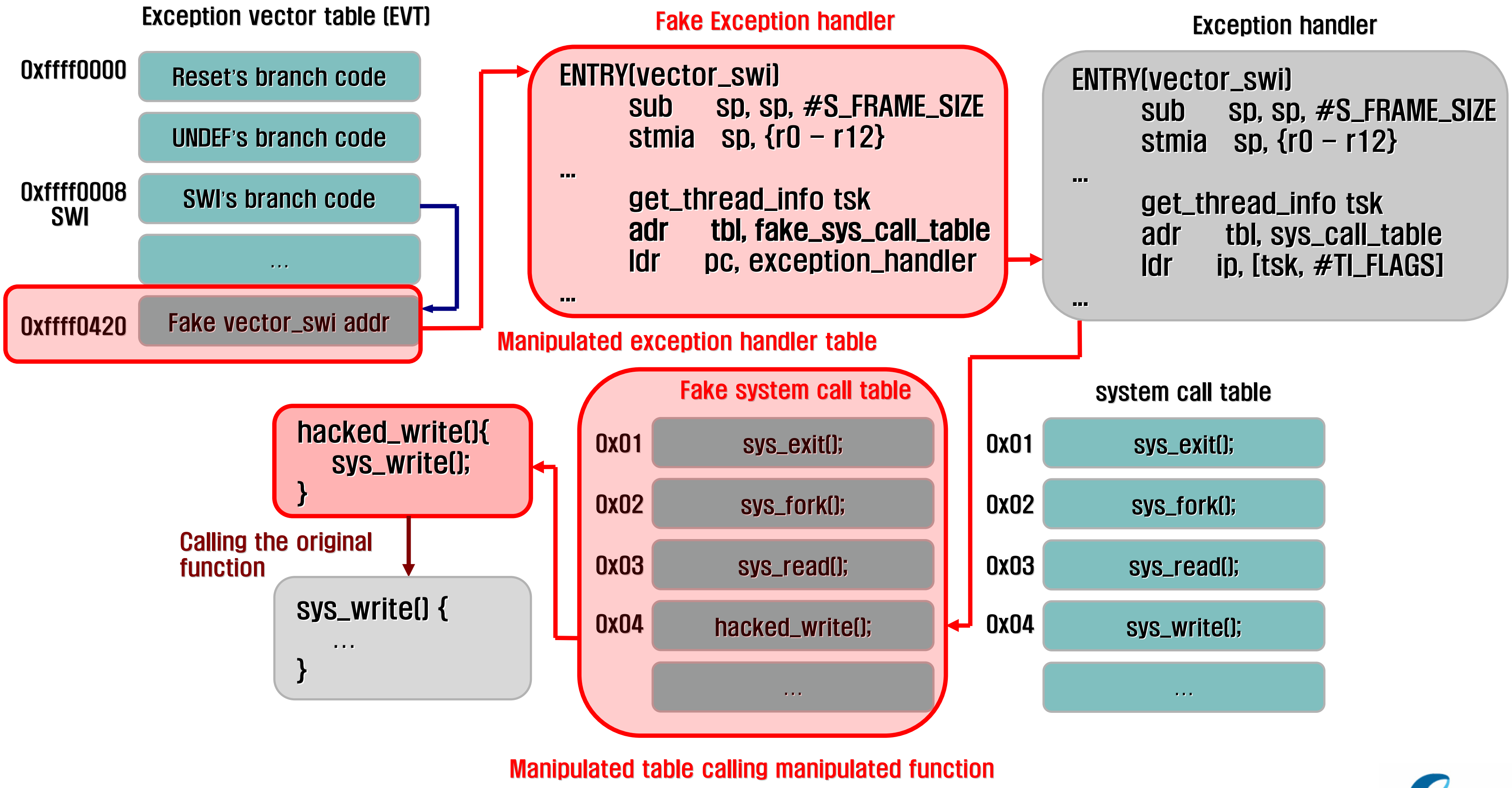
```
hacked_write(){
    sys_write();
}
```

```
sys_write() {
    …
}
```

Manipulated table calling manipulated function

Hooking Techniques 3/4

- EVT modifying hooking techniques (vector_swi handler)



Exception vector table (EVT)

0xffff0000	Reset's branch code
	UNDEF's branch code
0xffff0008 SWI	SWI's branch code
	…
0xffff0420	Fake vector_swi addr

Fake Exception handler

```
ENTRY(vector_swi)
        sub     sp, sp, #S_FRAME_SIZE
        stmia   sp, {r0 – r12}
…
        get_thread_info tsk
        adr     tbl, fake_sys_call_table
        ldr     pc, exception_handler
…
```

Exception handler

```
ENTRY(vector_swi)
        sub     sp, sp, #S_FRAME_SIZE
        stmia   sp, {r0 – r12}
…
        get_thread_info tsk
        adr     tbl, sys_call_table
        ldr     ip, [tsk, #TI_FLAGS]
…
```

Manipulated exception handler table

Fake system call table

0x01	sys_exit();
0x02	sys_fork();
0x03	sys_read();
0x04	hacked_write();
	…

system call table

0x01	sys_exit();
0x02	sys_fork();
0x03	sys_read();
0x04	sys_write();
	…

```
hacked_write(){
    sys_write();
}
```

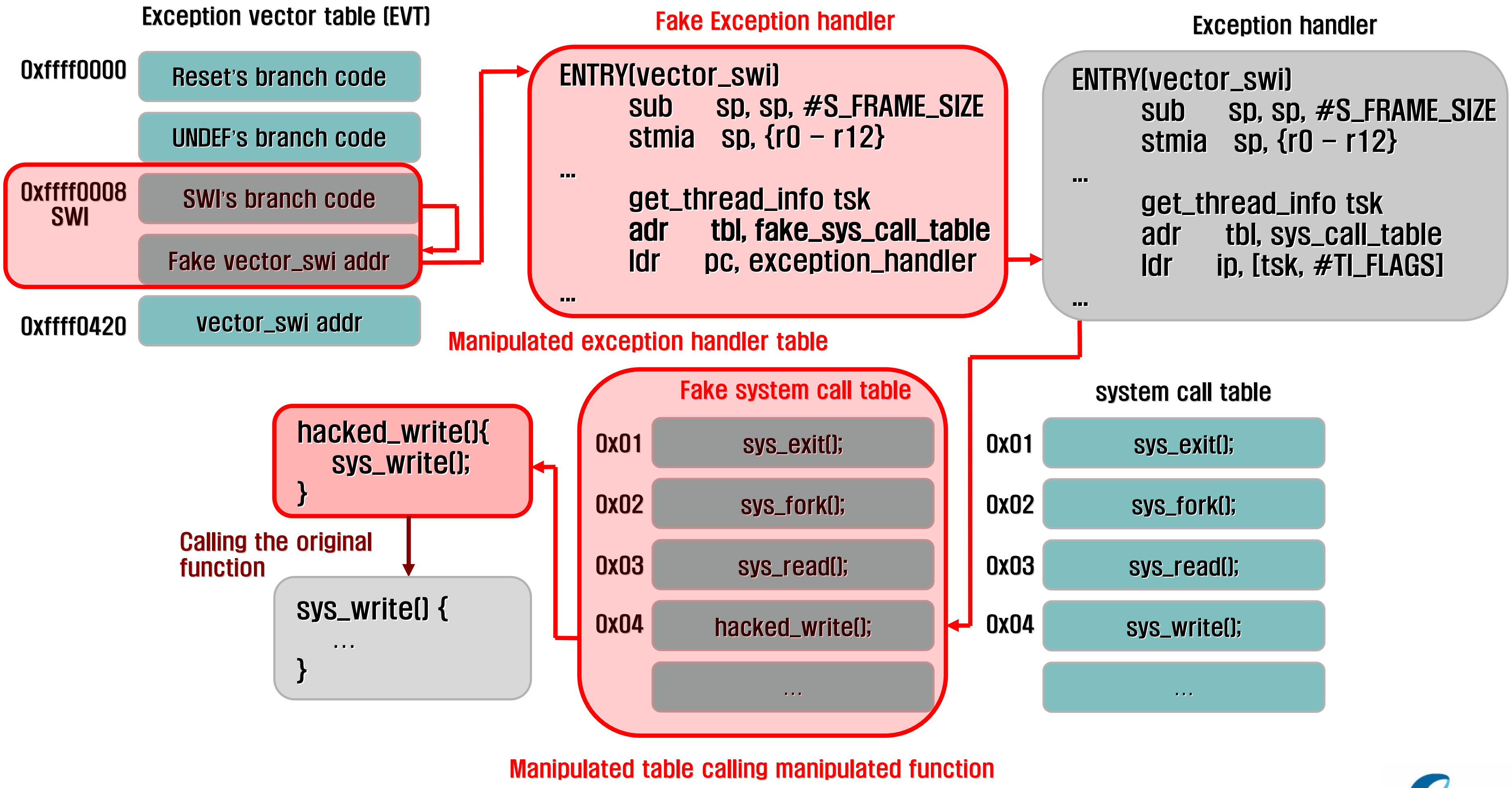
Calling the original function

```
sys_write() {
    …
}
```

Manipulated table calling manipulated function

Hooking Techniques 4/4

- EVT modifying hooking techniques (branch instruction offset)



Exception vector table (EVT)

0xffff0000	Reset's branch code
	UNDEF's branch code
0xffff0008 SWI	SWI's branch code
	Fake vector_swi addr
0xffff0420	vector_swi addr

Fake Exception handler

```
ENTRY(vector_swi)
        sub     sp, sp, #S_FRAME_SIZE
        stmia   sp, {r0 – r12}
…
        get_thread_info tsk
        adr      tbl, fake_sys_call_table
        ldr      pc, exception_handler
…
```

Exception handler

```
ENTRY(vector_swi)
        sub     sp, sp, #S_FRAME_SIZE
        stmia   sp, {r0 – r12}
…
        get_thread_info tsk
        adr      tbl, sys_call_table
        ldr      ip, [tsk, #TI_FLAGS]
…
```

Manipulated exception handler table

```
hacked_write(){
    sys_write();
}
```

Calling the original function

```
sys_write() {
   …
}
```

Fake system call table

0x01	sys_exit();
0x02	sys_fork();
0x03	sys_read();
0x04	hacked_write();
	…

system call table

0x01	sys_exit();
0x02	sys_fork();
0x03	sys_read();
0x04	sys_write();
	…

Manipulated table calling manipulated function

Demonstration

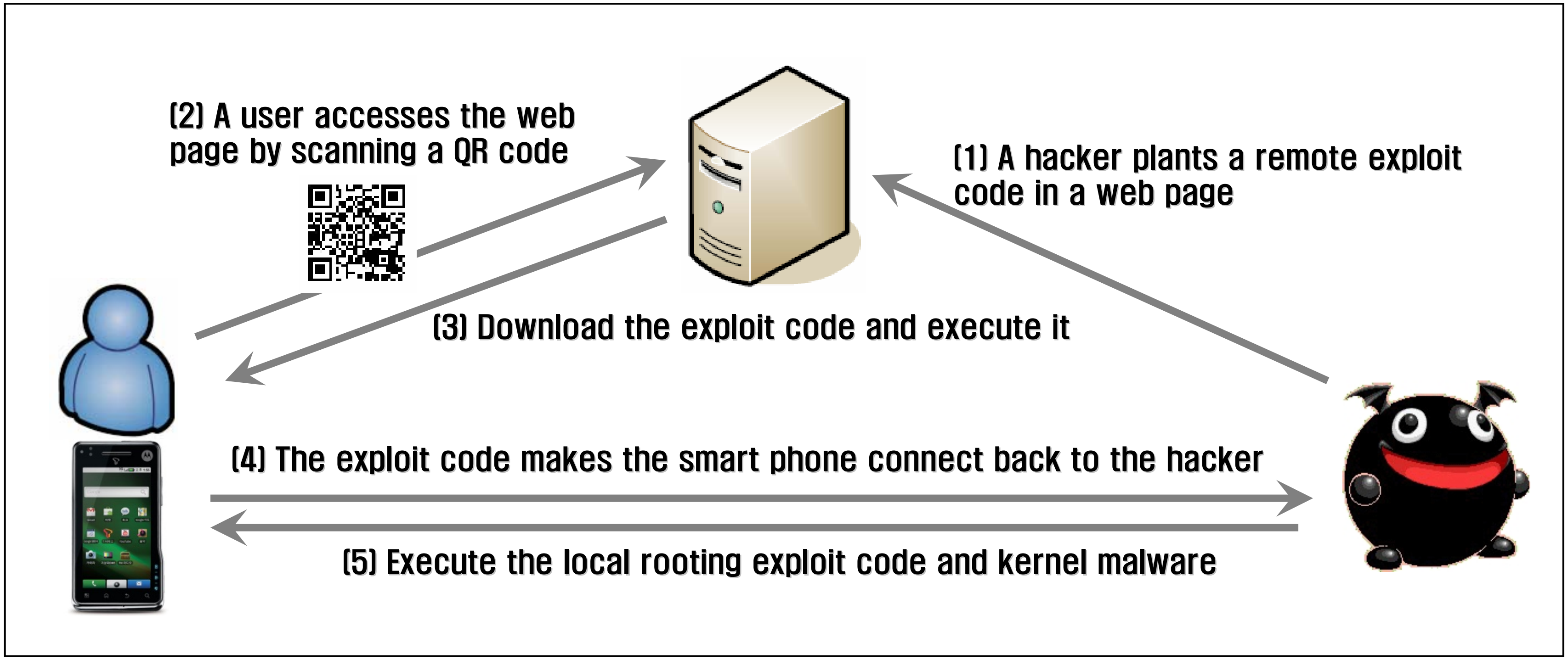
Android platform based kernel rootkit

Demonstration 1/3

- **Android platform based kernel rootkit test environment**
  - H/W: Motoroi XT720, S/W version: Android 2.1 (Eclair)
    Linux version 2.6.29-omap1 (w21679@zkr30mdb05) (gcc version 4.4.0 (GCC))
  - H/W: Galaxy S, Galaxy tab, S/W version: Android 2.2 (Froyo)
    Linux version 2.6.32.9 (root@SEI-27) (gcc version 4.4.1 (Sourcery G++ Lite 2009q3-67))
  - H/W: Optimus one, S/W version: Android 2.2 (Froyo)
    Linux version 2.6.32.9 (mclab1@s-ibm06-desktop) (gcc version 4.4.0 (GCC))

Demonstration 2/3

- **Android Remote / Local exploitation**
  - Acquiring a shellcode and rooting via QR code scanning
  - Webkit library use-after-free vulns (drive-by download)
  - CVE-2009-3547 linux kernel local pipe vulnerability



(2) A user accesses the web page by scanning a QR code

(1) A hacker plants a remote exploit code in a web page

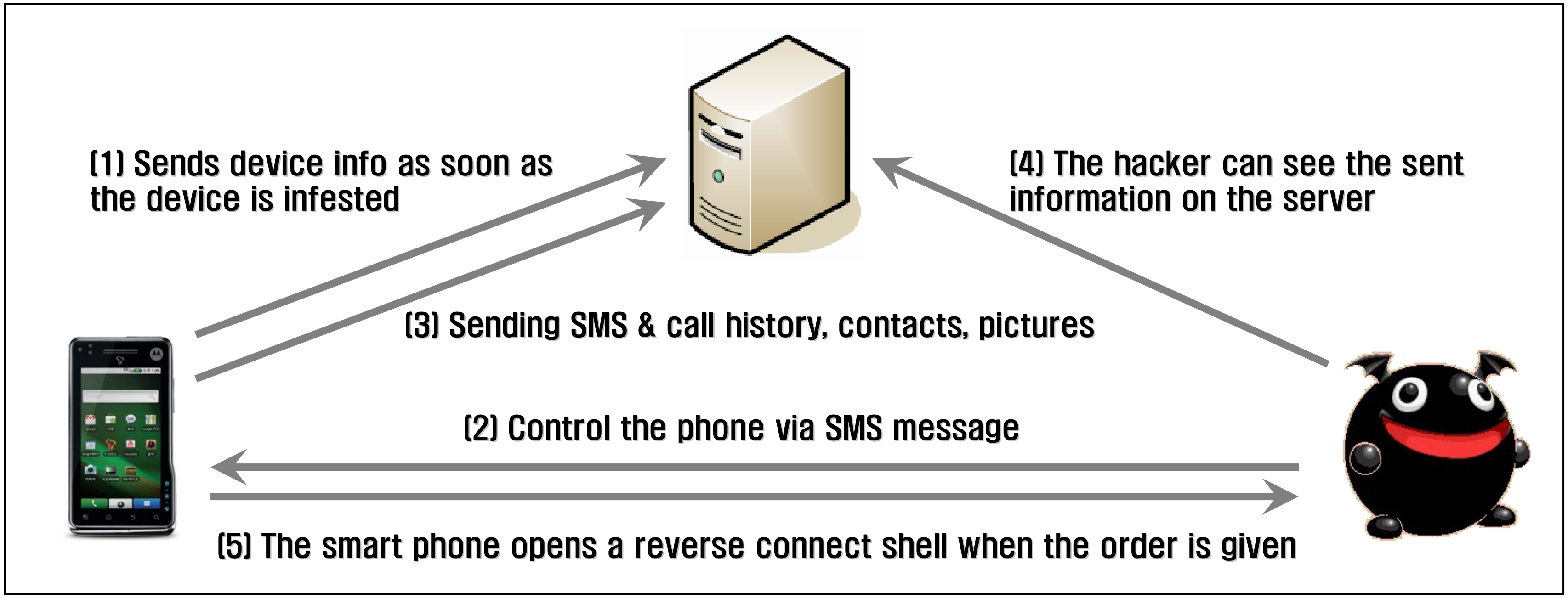
(3) Download the exploit code and execute it

(4) The exploit code makes the smart phone connect back to the hacker

(5) Execute the local rooting exploit code and kernel malware

Demonstration 3/3

- **Android kernel rootkit**
  - Enabling kernel malwares and acquiring critical information (device info, history of SMS and calls, contacts, pictures, GPS info)
  - Hiding file & directory, process, LKM module driver
  - remote reverse shell connection



(1) Sends device info as soon as the device is infested

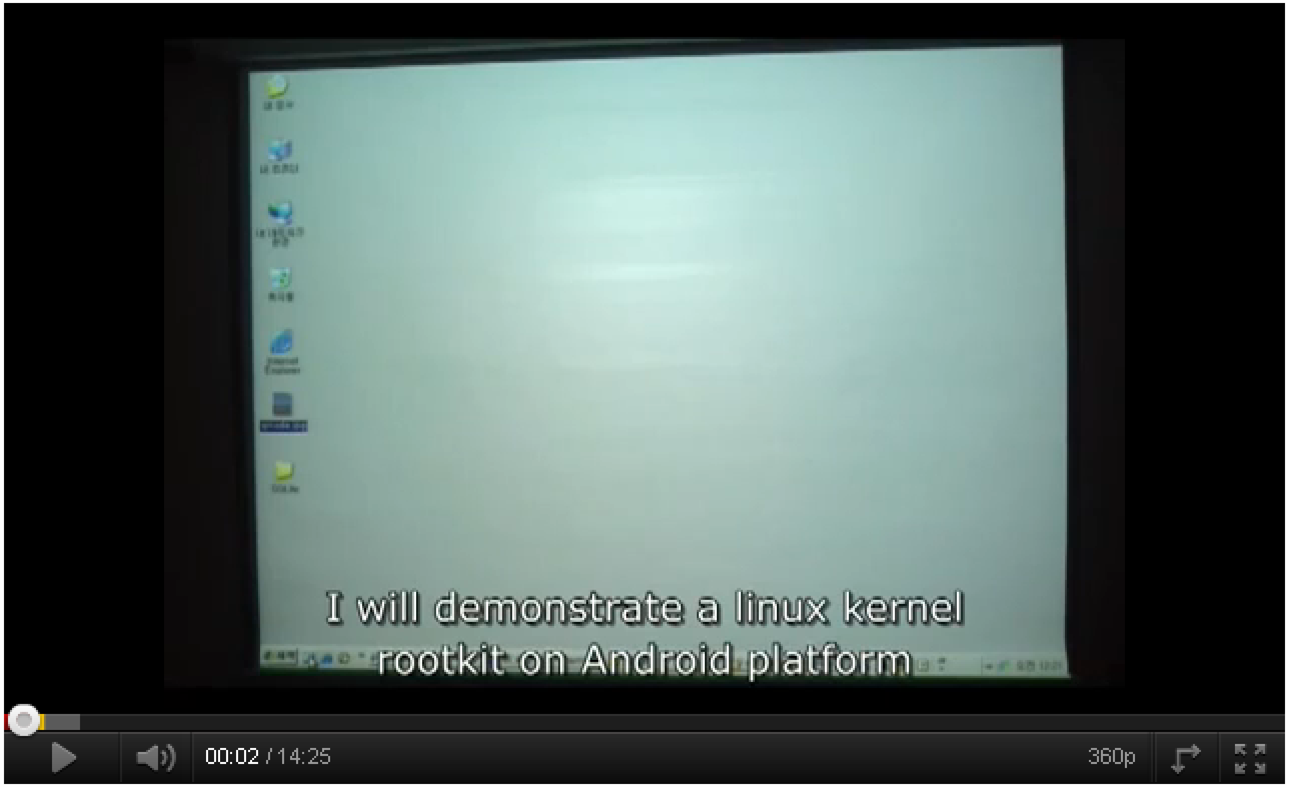
(4) The hacker can see the sent information on the server

(3) Sending SMS & call history, contacts, pictures

(2) Control the phone via SMS message

(5) The smart phone opens a reverse connect shell when the order is given

Demonstration of Android Kernel Rootkit



I will demonstrate a linux kernel rootkit on Android platform

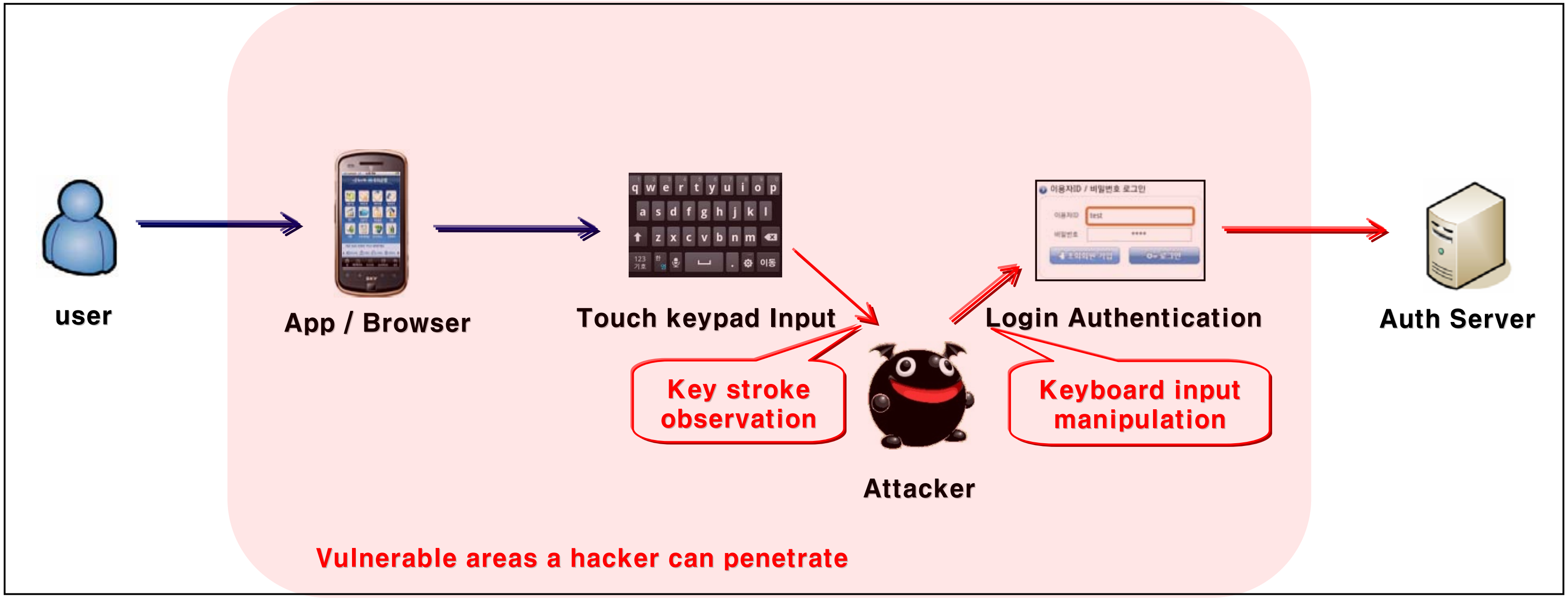
http://www.youtube.com/watch?v=HZ8J-manvPk

Conclusion

Effect of smart platform rootkit

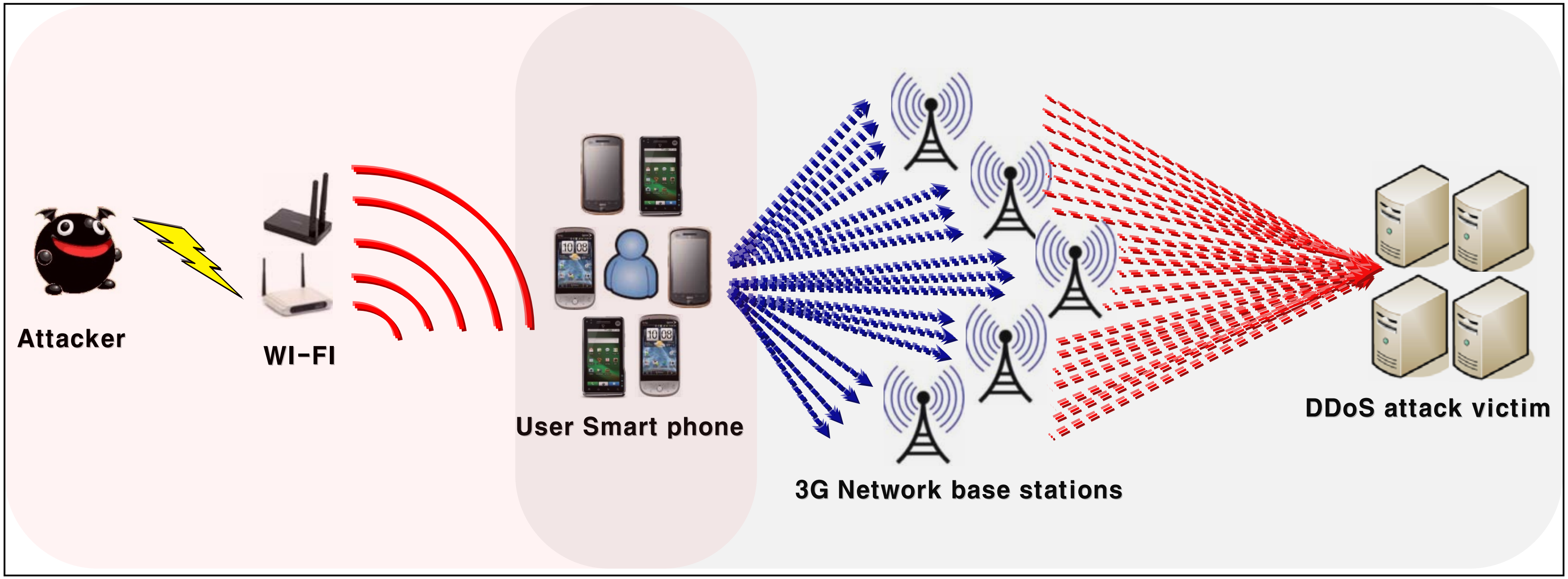
Effect of smart phone rootkit 1/2

- Future threats for the Android platform
  - – Touchpad keylogging (interrupt hooking method)
  - – Internet banking transaction manipulation



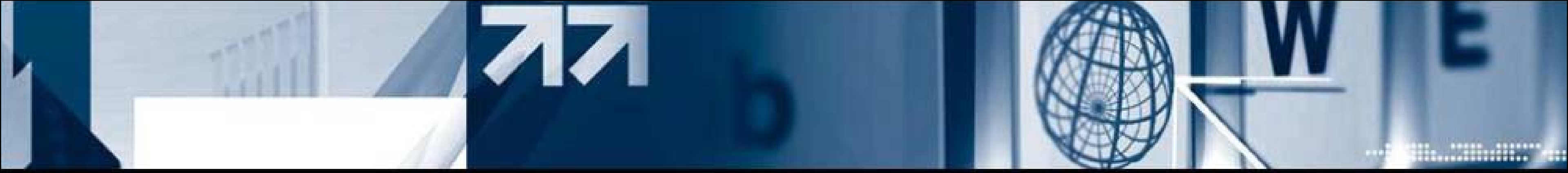
Effect of smart phone rootkit 2/2

- Future threats for the Android platform
  - Advanced kernel based botnet (conceal C&C tools and connection channels)
  - Kernel rootkit that hides the malwares



Conclusion

- Future works
  - – Various kernel based rootkits for various smart platform
  - – Detecting manipulated kernel memory and hidden malwares
  - – Kernel protection mechanism for kernel integrity
  - – Building a fundamental security policy for smart platform



Question?

Contact info: "dong-hoon You" (Xpl017Elz), in INetCop(c).
E-mail: x82(at)inetcop(dot)org
Home: http://x82.inetcop.org