

---

# Android smart platform kernel malware

## Android platform based linux kernel rootkit

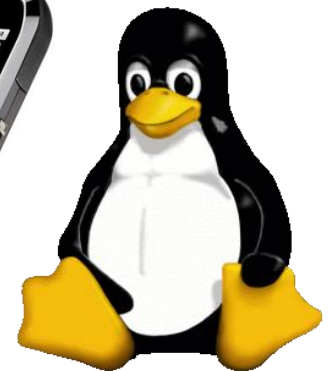
---

2011. 11. 03.

Dong-Hoon You  
INetCop Security  
x82@inetcop.org

# Contents

- Android smart platform kernel malware
  - Introduction
  - Technical Description I
  - Technical Description II
  - Demonstration



# Introduction

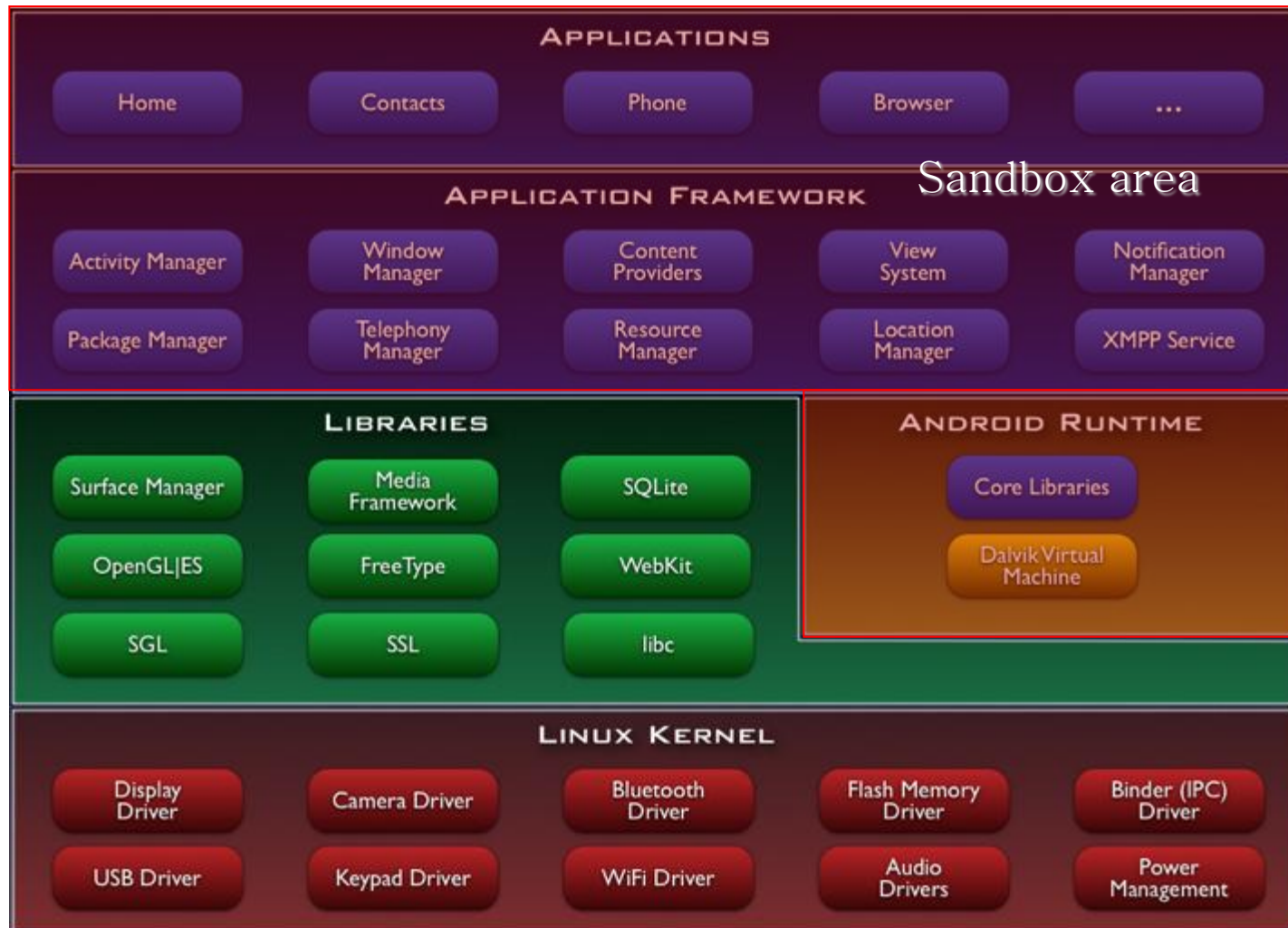
Android smart platform sandbox



## The purpose of Sandbox

The ideal operating of a sandbox

- Android smart platform sandbox

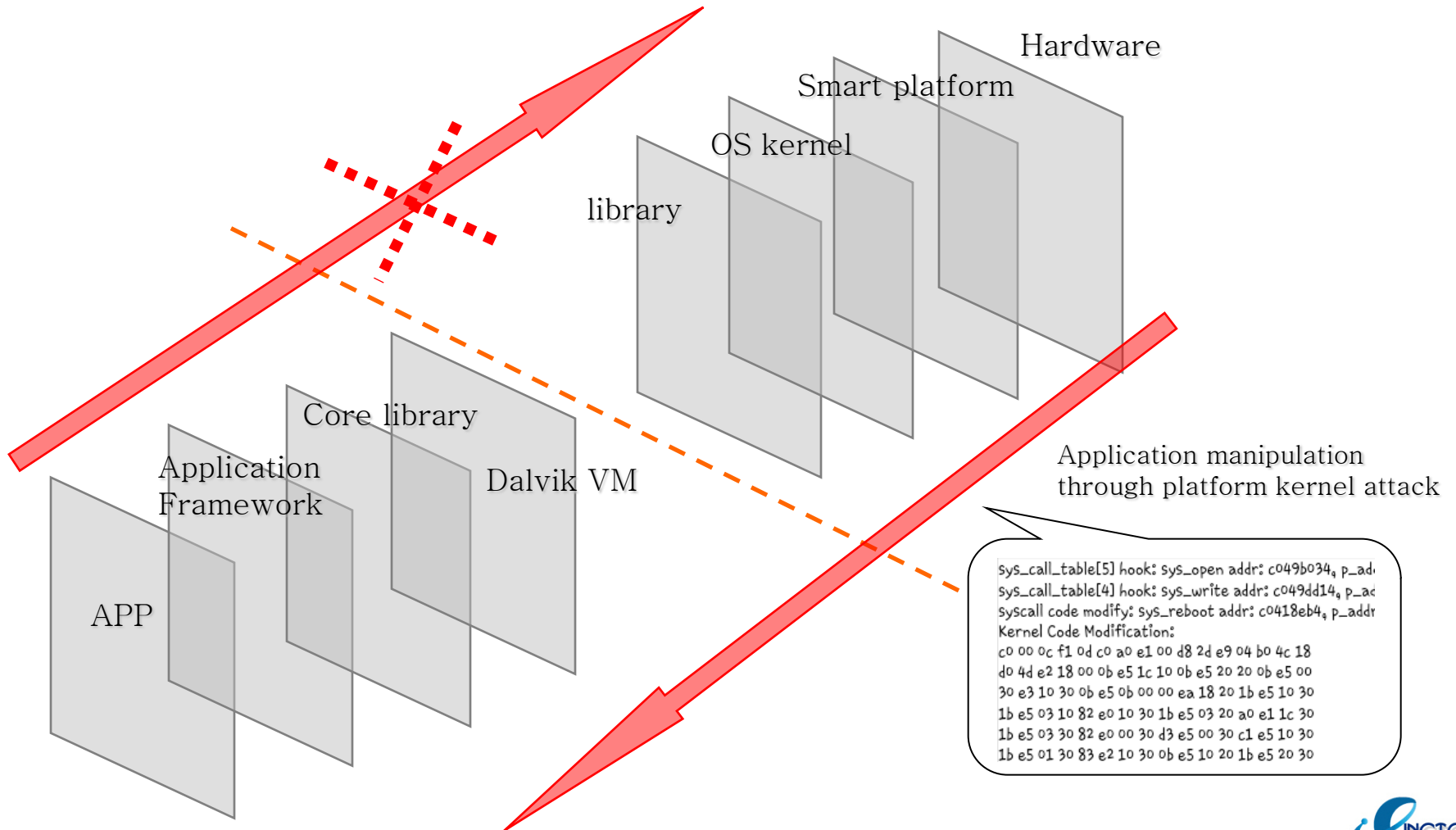




## Reality of sandbox security

However, the reality is a battlefield

- Security Technology Issue



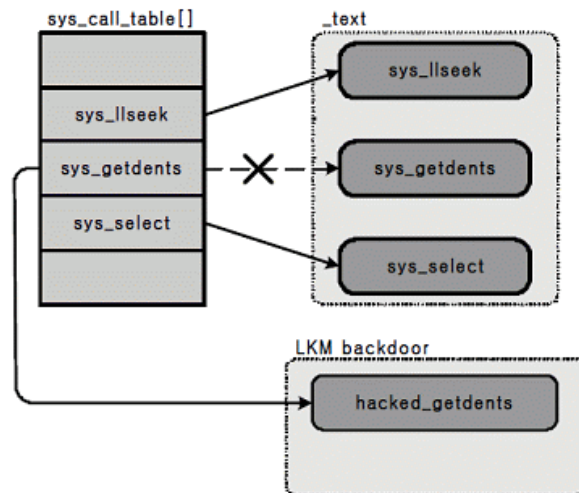
- **Security threat caused by malware inside of Sandbox**
  - various Backdoor, Trojan S/W and malwares with a form of app
  - Overcharged fee, personal information leak, eavesdropping etc.
  - H/W device meltdown, sending spam SMS, DDoS botnet attack
  - Can be handled by verifying app integrity and a vaccine
  
- **Security threat caused by vulnerabilities out side of Sandbox**
  - Android 3<sup>rd</sup> party application and web browser remote attack , 2010
  - Local rooting exploit code using kernel vulnerability, Aug 2009
  - LKM type kernel rootkit (Defcon 18) , June, 2010
  - Hard to apply a security update on a smart platform

\* Reference: <http://www.exploit-db.com>

# Technical Description I

Kernel Level Hooking Techniques

- 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



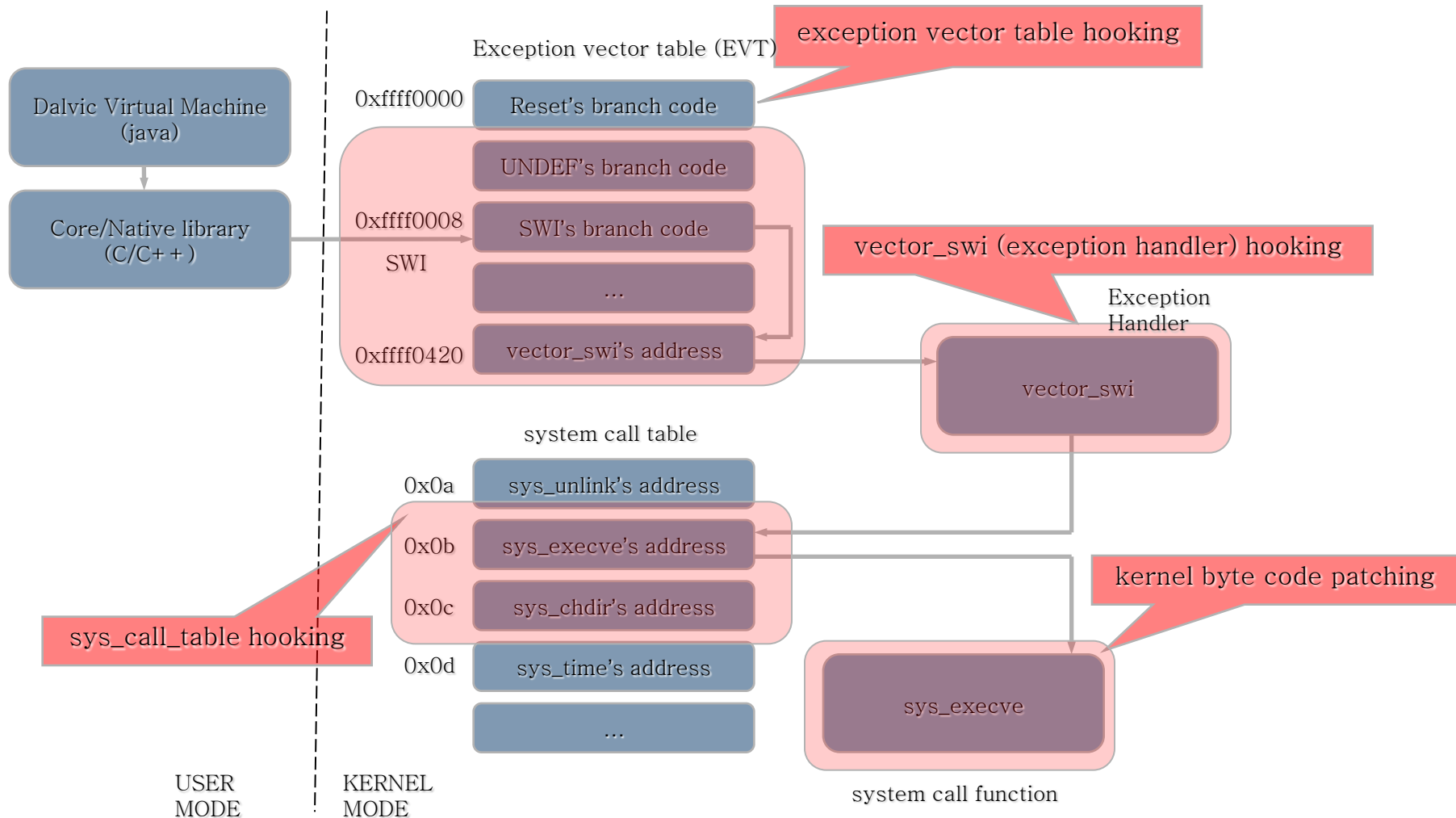
```
$ export PATH=/data/local/bin:$PATH
$ lsmod
pcnet32 28744 0 - Live 0xd0cb9000
btusb 10316 0 - Live 0xd0c48000
sco 8948 0 - Live 0xd0c21000
rfcomm 29876 0 - Live 0xd0a71000
bnep 10976 0 - Live 0xd0a27000
l2cap 19444 4 rfcomm,bnep, Live 0xd09f6000
bluetooth 47784 5 btusb,sco,rfcomm,bnep,l2cap, Live 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
$
```

- 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 module binary

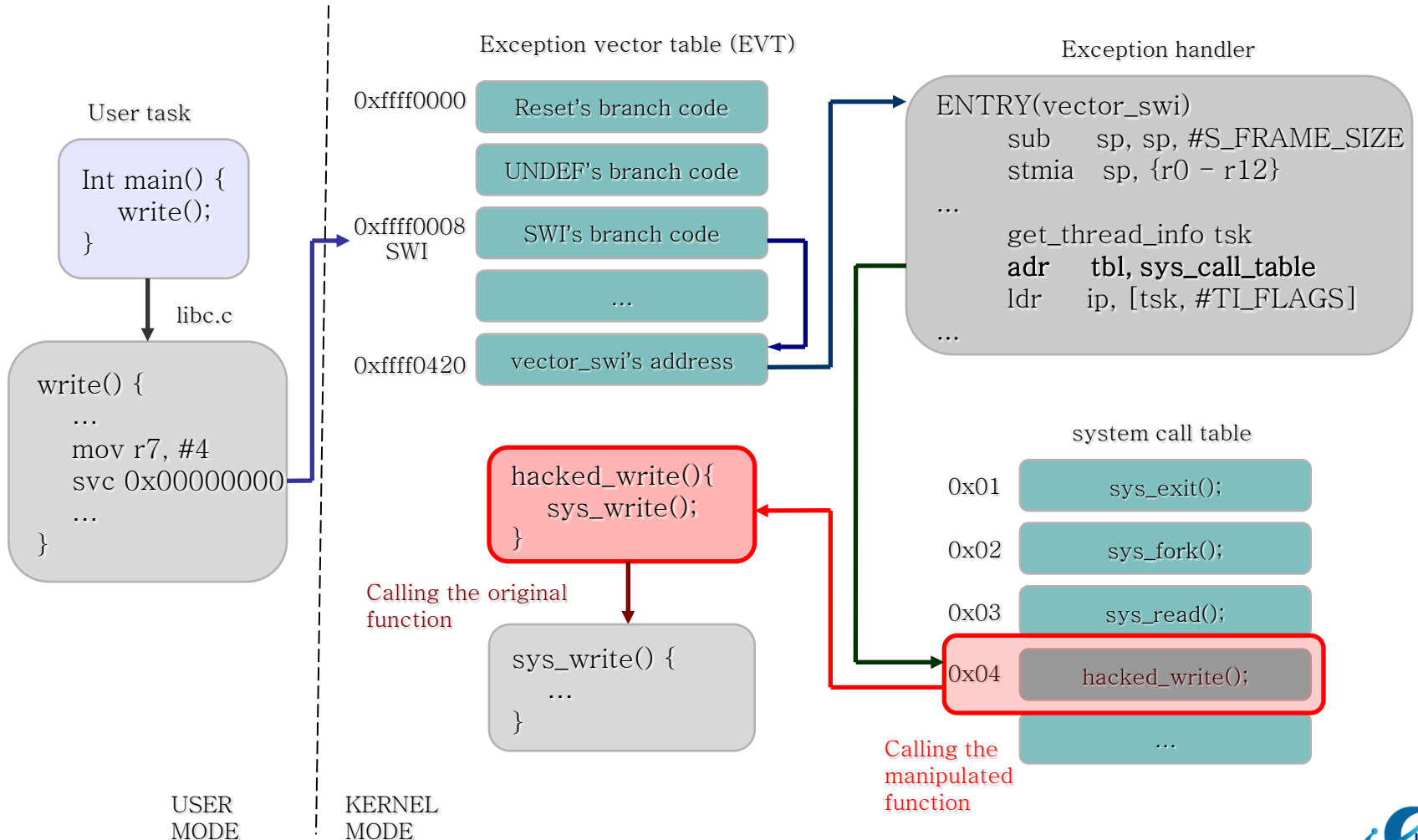
```
fs/open.c:
EXPORT_SYMBOL(sys_close);
...
call.S:
/* 0 */      CALL(sys_restart_syscall)
             CALL(sys_exit)
             CALL(sys_fork_wrapper)
             CALL(sys_read)
             CALL(sys_write)
/* 5 */      CALL(sys_open)
             CALL(sys_close)
```

```
# 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 '
#
```

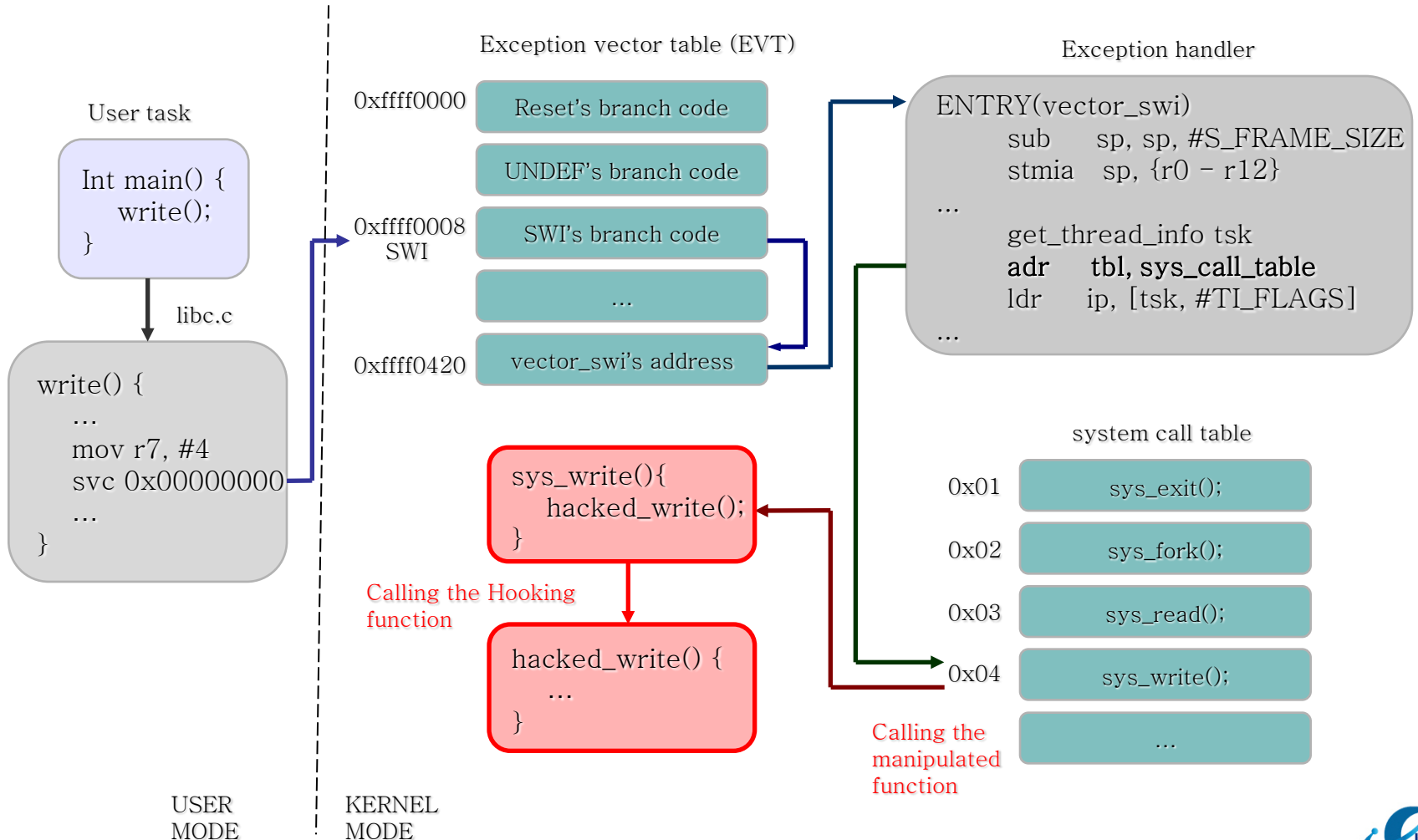
- Branches of Android platform kernel hooking



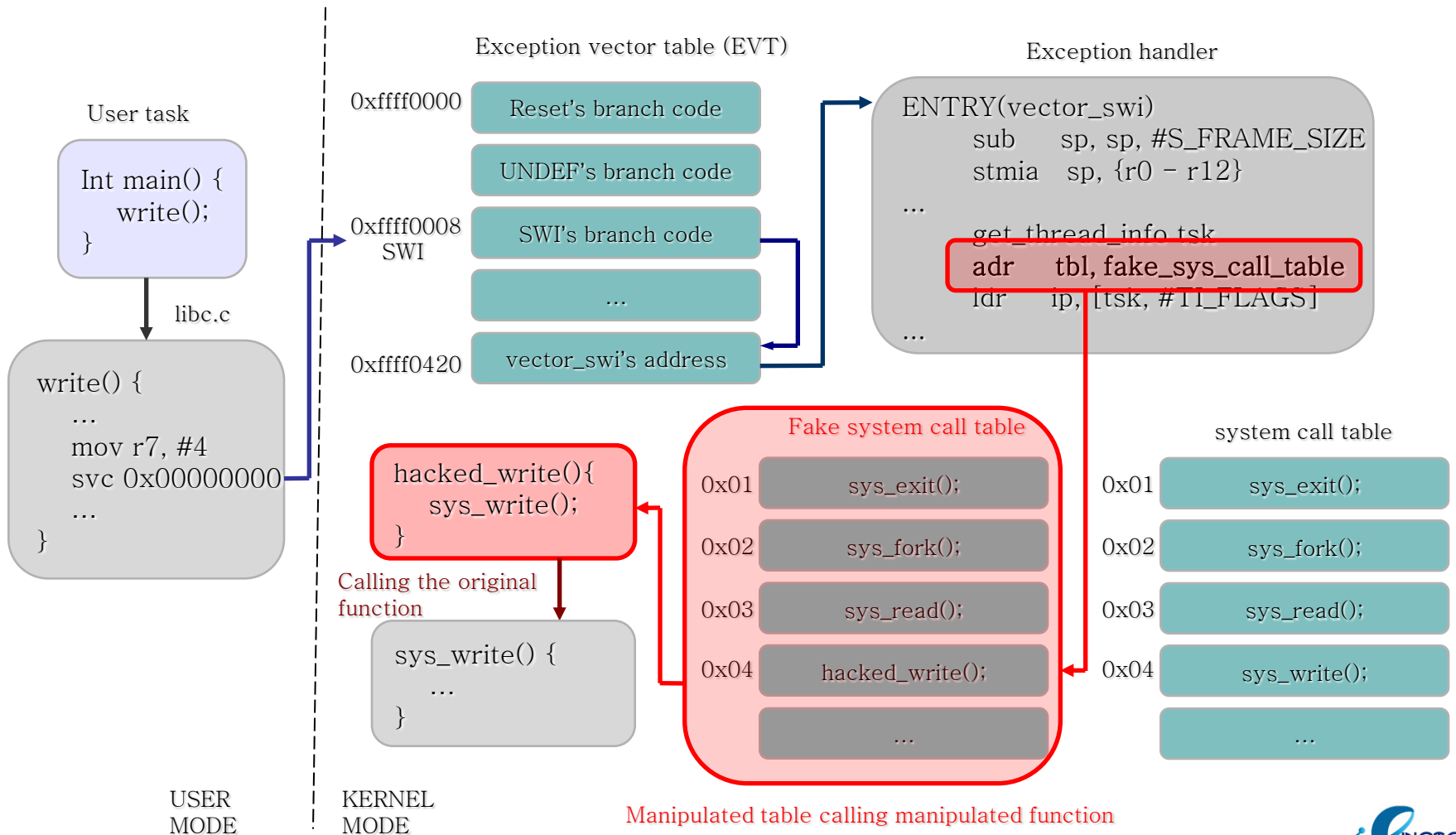
## Basic techniques for sys\_call\_table hooking #1



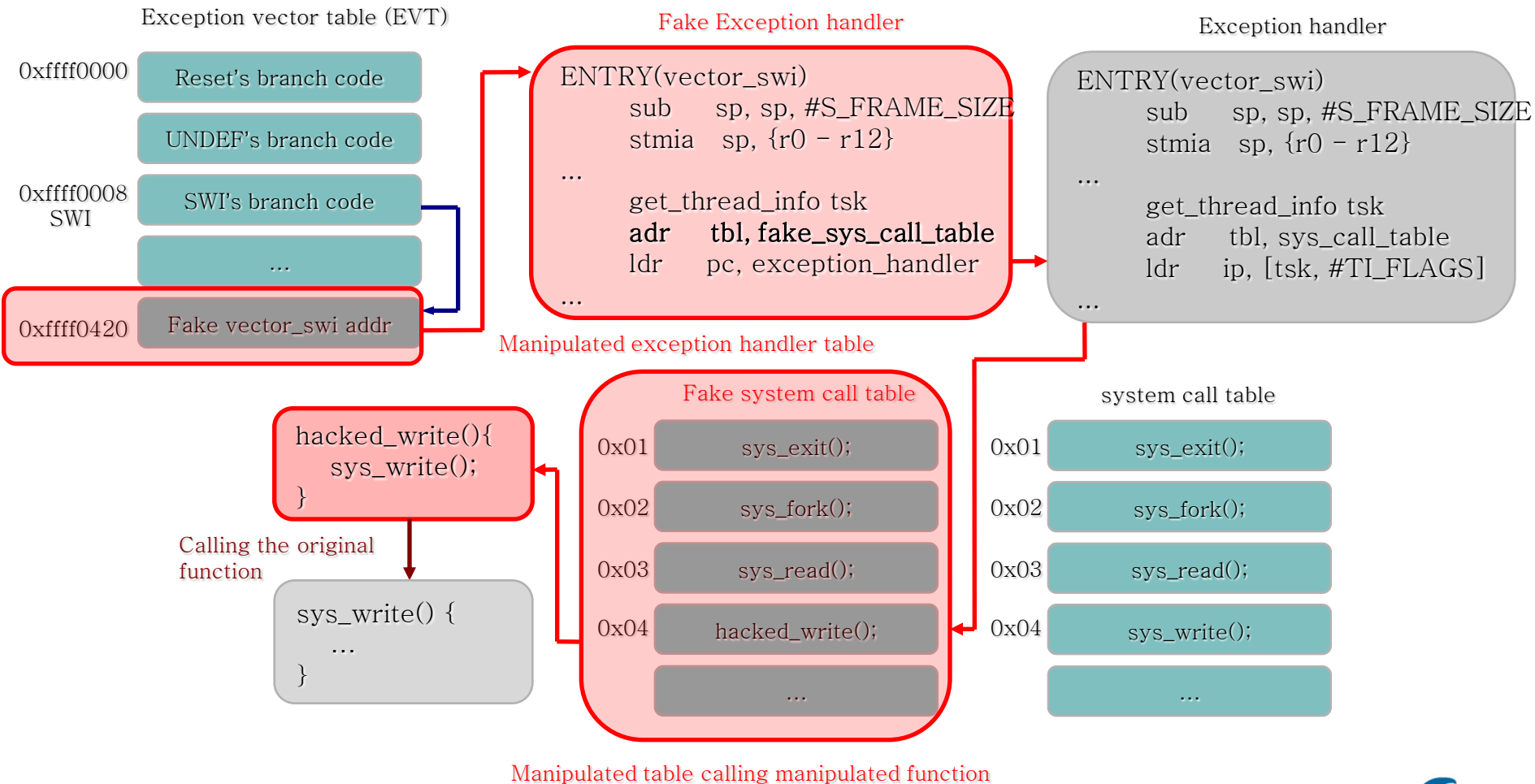
## Basic techniques for sys\_call\_table hooking #2



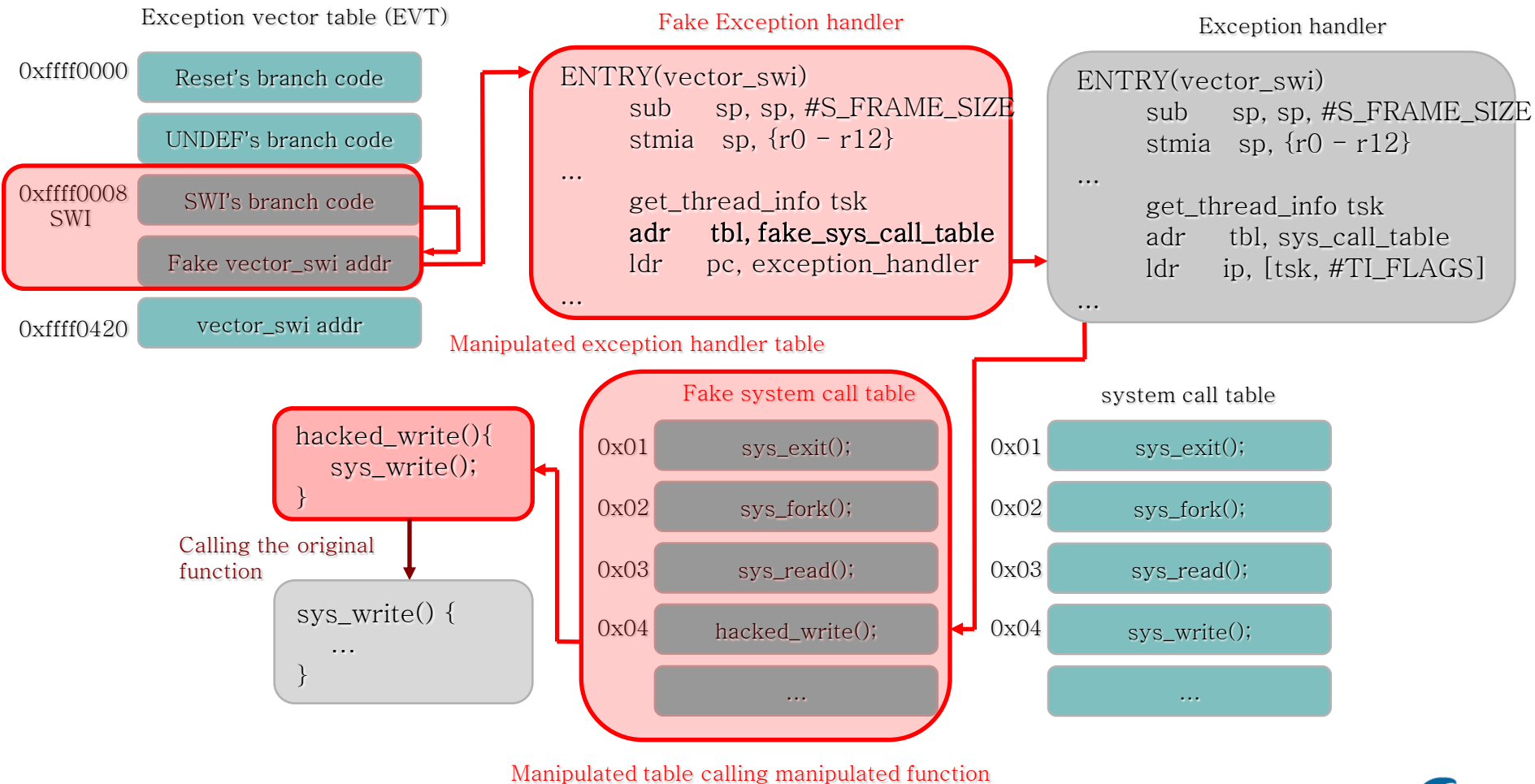
## ■ Modifying vector\_swi handler routine



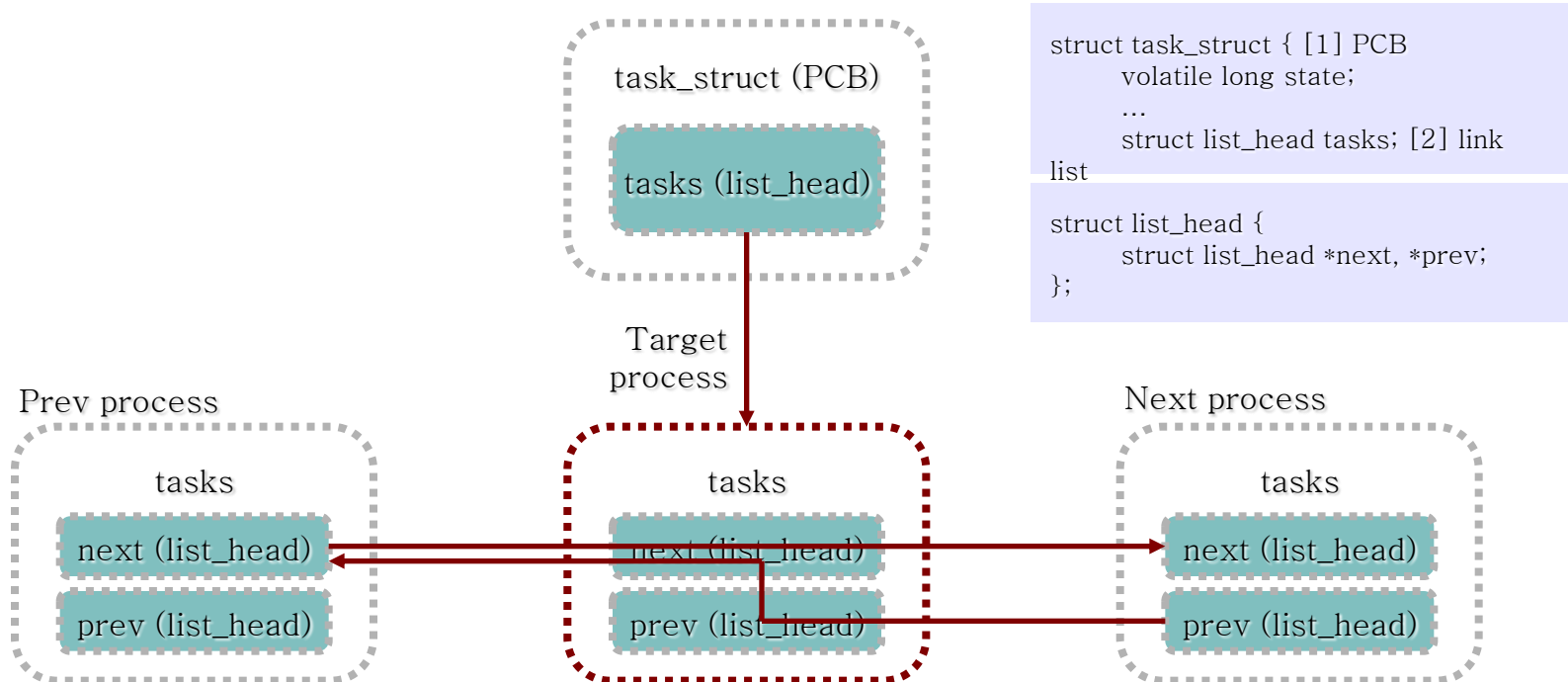
## EVT modifying hooking techniques (vector\_swi handler)



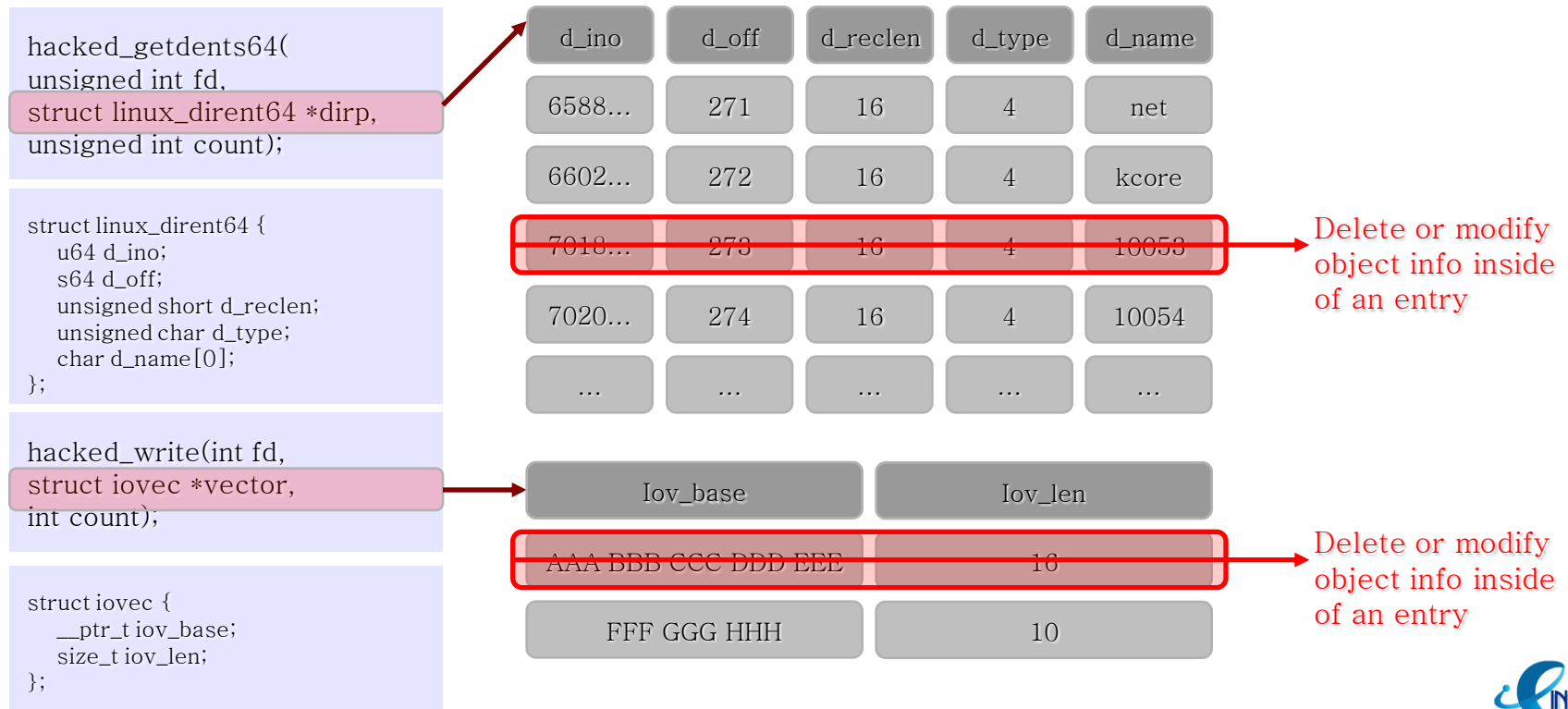
- EVT modifying hooking techniques (branch instruction offset)



- How the Android kernel malwares hide their processes
- Basic of Process hiding
  - Modifying list\_head structure in PCB structure holding process info.
  - Combine prev(information of previous process) and next (next process)



- How Android kernel malwares hide their resources
- Basic of File & directory & Network status hiding
  - Hook system calls by modifying EVT, vector\_swi, sys\_call\_table
  - Delete object info in dirent64 structure after hooking getdents64
  - Delete or modify network info after hooking write, writev function



- Basic information hiding method of Android kernel malwares
- Kernel module driver hiding
  - Modifying list\_head structure inside of module structure holding information about loaded module
  - Combine prev(information of previous process) and next (next process)

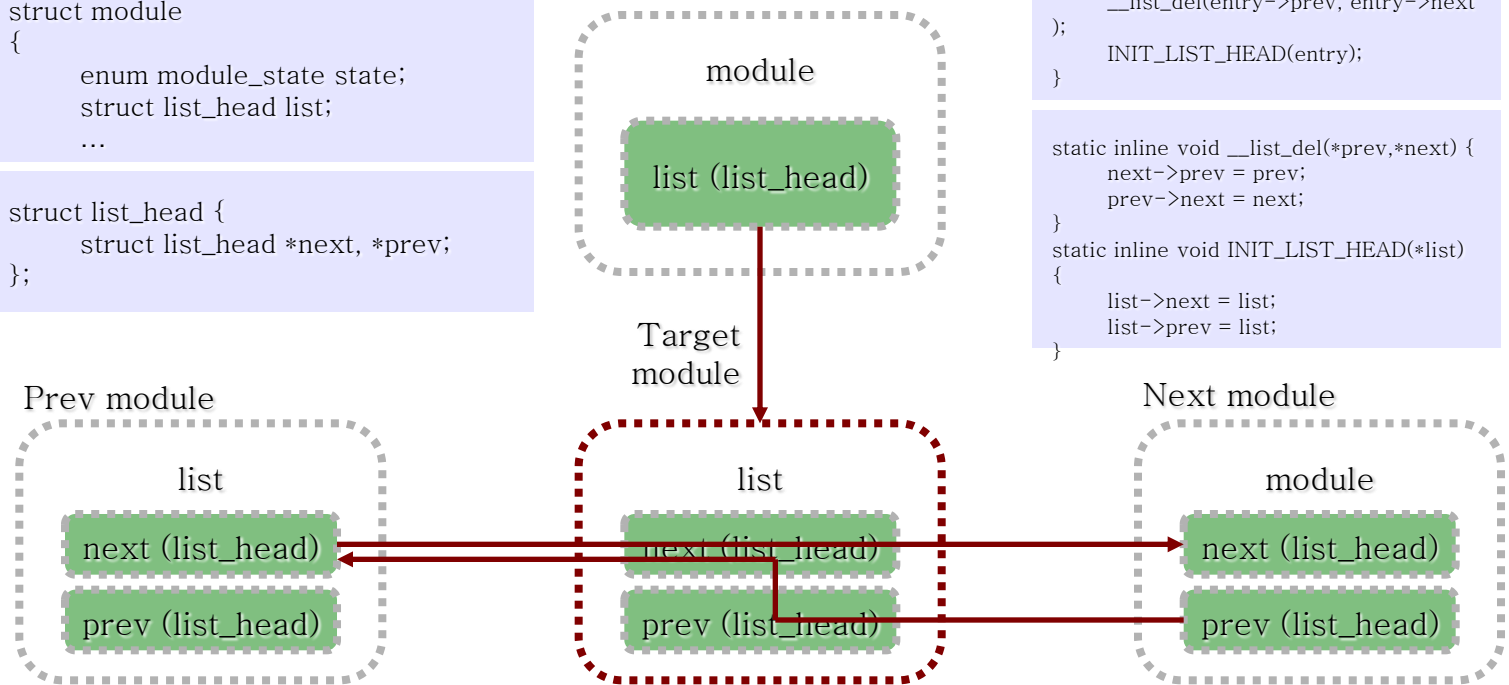
```
struct module
{
    enum module_state state;
    struct list_head list;
    ...
};

struct list_head {
    struct list_head *next, *prev;
};
```

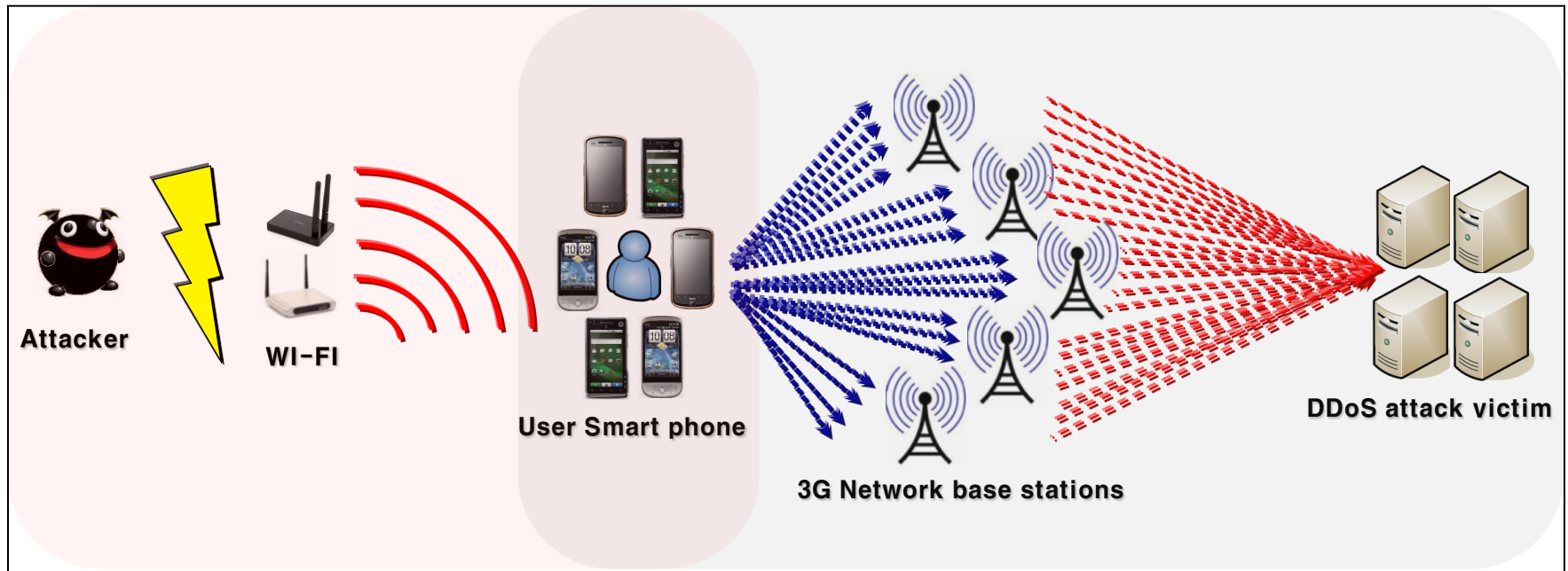
```
static inline void list_del_init(*entry) {
    __list_del(entry->prev, entry->next);
};
INIT_LIST_HEAD(entry);

static inline void __list_del(*prev,*next) {
    next->prev = prev;
    prev->next = next;
}

static inline void INIT_LIST_HEAD(*list)
{
    list->next = list;
    list->prev = list;
}
```



- Future threats for the Android platform
  - Touchpad keylogging (interrupt hooking method)
  - Internet banking transaction manipulation
  - Advanced kernel based botnet (conceal C&C tools and connection channels)
  - Kernel rootkit that hides the malwares

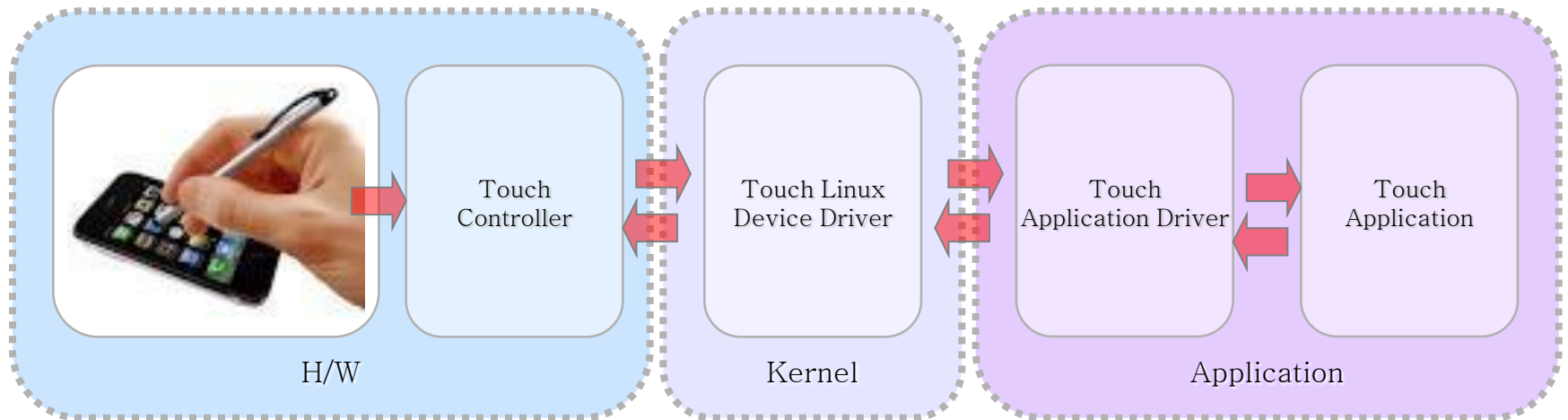


# Technical Description II

Input device Hooking Touchpad Key  
logger

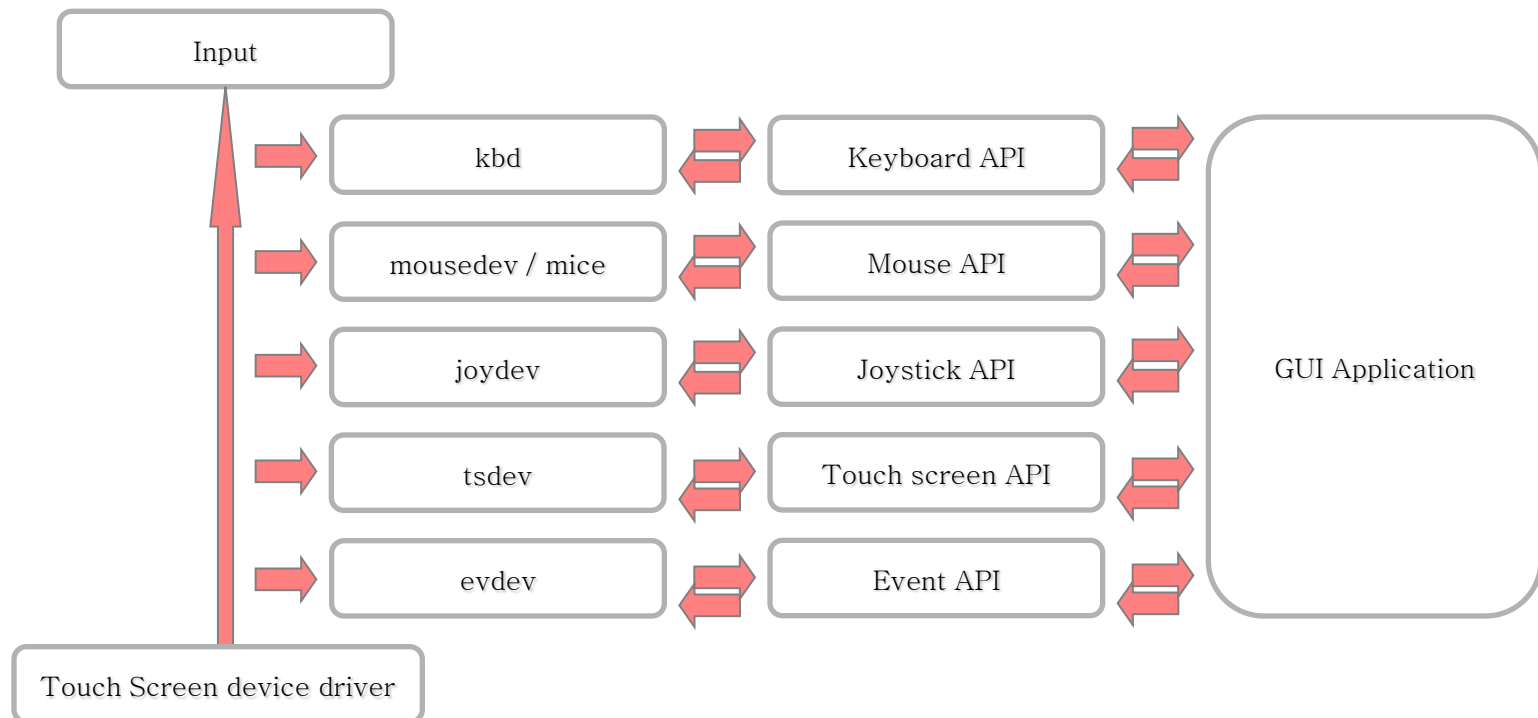
- System structure for input process

- Hardware
  - touch pad controller
- Kernel device controller
  - interpret user input and forward it to application
  - Since kernel 2.6, consolidated into a “input device controller”
- Application
  - Consists of library modules and applications that use the libraries



## ■ Input device driver

- Mouse, touch screen, joystick, keyboard and so on.
- integrated input device driver for Linux to handle various devices
- Read input information and make the applications react to the info.
- Device driver handling method is standardized to handle various device



- Input device driver

- Provide registered event handler info and input device info
- Connect input device driver and event handler device
- Forward input to each event handler

```
ls -l /proc/bus/input
-r--r--r-- root    root          0 2011-10-19 15:05 handlers
-r--r--r-- root    root          0 2011-10-19 15:05 devices
# cat /proc/bus/input/handlers
cat /proc/bus/input/handlers
N: Number=0 Name=evdev Minor=64
```

```
I: Bus=0000 Vendor=0000 Product=0000 Version=0000
N: Name="qt602240_ts_input"
P: Phys=
S: Sysfs=/devices/virtual/input/input3
U: Uniq=
H: Handlers=mouse0 event3
B: EV=b
B: KEY=400 0 0 0 0 0 0 0 0 0 0
B: ABS=650000 11000003
```

- **Event handler device driver**
  - Actual device driver file to read input from input devices
  - Save input value into internal buffer after transform it properly
  - Make input value readable to application
  
- **evdev event handler device driver**
  - Device driver that reads in input value in general type
  - Each driver for each device, so it can read all the device values
  - All the drivers have same architecture, independent from H/W
  - Forward input events from kernel to application
  - Send time, type, key value in struct `input_event` structure type
  - Realized in `linux/drivers/input/evdev.c`
  - Exists as an string device type in path `‘/dev/input/event*’`

- Touch Keypad input\_event structure type
  - time: when the input event occurs
  - type: the type of input device (ex: touch screen and mouse)
    - Touch pad uses absolute value (ABS\_X, ABS\_Y) for location
    - Mouse uses amount of change (REL\_X, REL\_Y) for location
  - code: Categorize input information
    - Member value to discern input information
  - value: input value for each code member
    - for touch screen and mouse, input location is stored

```
ls -l /dev/input/  
crw-rw---- root    input    13,   71 2011-10-19 16:48 event7  
crw-rw---- root    input    13,   70 2011-10-16 07:44 event6  
crw-rw---- root    input    13,   69 2011-10-16 07:44 event5  
crw-rw---- root    input    13,   68 2011-10-16 07:44 event4  
crw-rw---- root    input    13,   67 2011-10-16 07:44 event3  
crw-rw---- root    input    13,   66 2011-10-16 07:44 event2  
crw-rw---- root    input    13,   65 2011-10-16 07:44 event1  
crw-rw---- root    input    13,   64 2011-10-16 07:44 event0
```

```
struct input_event{  
    struct timeval    time;  
    unsigned short    type;  
    unsigned short    code;  
    unsigned int      value;  
};
```

- Contents of Touch Keypad input\_event structure

```
# hexdump -C /dev/input/event3
...
00000060 2b 70 00 00 b2 97 04 00 03 00 35 00 9a 00 00 00 |tp.....5....|
00000070 2b 70 00 00 cb 97 04 00 03 00 36 00 cb 01 00 00 |tp.....6....|
00000080 2b 70 00 00 d2 97 04 00 03 00 30 00 00 00 00 00 |tp.....0....|
00000090 2b 70 00 00 d9 97 04 00 03 00 32 00 02 00 00 00 |tp.....2....|
000000a0 2b 70 00 00 e0 97 04 00 00 00 02 00 00 00 00 00 |tp.....|
000000b0 2b 70 00 00 eb 97 04 00 00 00 00 00 00 00 00 00 |tp.....|
```

```
linux/include/linux/input.h:
struct input_event {
    struct timeval time; // 2b 70 00 00 (tv_sec) cb 97 04 00 (tv_usec)
    unsigned short type; // 03 00
    unsigned short code; // 36 00
    unsigned int value; // cb 01 00 00
};
```

```
defined in linux/include/linux/input.h
...
#define EV_SYN 0x00 // synchronize
#define EV_KEY 0x01 // key or button
#define EV_ABS 0x03 // Absolute
coordinate
...

```

type	code	value	Description
EV_KEY	BTN_TOUCH	0x0	Off from touch screen
EV_KEY	BTN_TOUCH	0x1	On the touch screen
EV_KEY	BTN_TOUCH	0x2	Consecutive touch
EV_ABS	ABS_X	X num	Absolute X coordinate
EV_ABS	ABS_Y	Y num	Absolute Y coordinate
EV_SYN	0x0	0x0	Event occurred

## Touch Keypad value logger



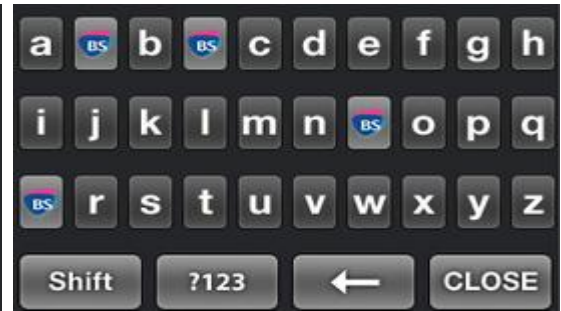
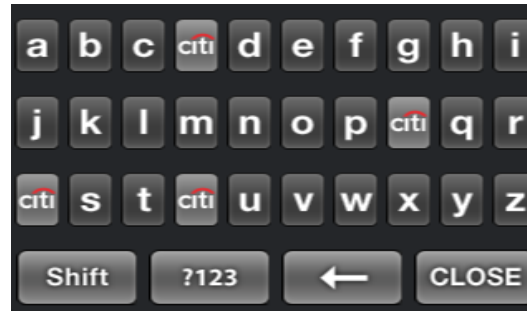
## ▪ Touch Keypad value Injection

- When a certain value is input, change it into other data
- evdev event handler can inject an event
- Using write function, write on event device file to inject event.
- the information should be struct\_input\_event structre type

```
unsigned char backspace[]={
0x81, 0x01, 0x00, 0x00, 0x31, 0x74, 0x05, 0x00, 0x03, 0x00, 0x35, 0x00,
0xae, 0x01, 0x00, 0x00, 0x81, 0x01, 0x00, 0x00, 0xb9, 0x74, 0x05, 0x00,
0x03, 0x00, 0x36, 0x00, 0xcc, unsigned char h_code[]={
0xef, 0x74, 0x05, 0x00, 0x03, 0x8f, 0x34, 0x02, 0x00, 0x1f, 0xdd, 0x03, 0x00, 0x03, 0x00, 0x35, 0x00,
0x81, 0x01, 0x00, 0x00, 0x23, 0x2b, 0x01, 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00, 0xaa, 0xdd, 0x03, 0x00,
0x04, 0x00, 0x00, 0x00, 0x81, 0x03, 0x00, 0x36, 0x00, 0x4b, 0x02, 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00,
0x00, 0x00, 0x02, 0x00, 0x00, 0xe0, 0xdd, 0x03, 0x00, 0x03, 0x00, 0x30, 0x00, 0x28, 0x00, 0x00, 0x00,
0x99, 0x75, 0x05, 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00, 0x14, 0xde, 0x03, 0x00, 0x03, 0x00, 0x32, 0x00,
0x81, 0x01, 0x00, 0x00, 0x5f, 0x04, 0x00, 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00, 0x48, 0xde, 0x03, 0x00,
0xae, 0x01, 0x00, 0x00, 0x81, usleep(10000); 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00,
0x03, 0x00, 0x36, 0x00, 0xcc, write(temp_fd,h_code,sizeof(h_code)); 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7e, 0x6f, 0x07, 0x00, 0x03, usleep(10000); 0x04, 0x00, 0x03, 0x00, 0x35, 0x00,
0x81, 0x01, 0x00, 0x00, 0x85, write(temp_fd,e_code,sizeof(e_code)); 0x02, 0x00, 0x91, 0x85, 0x04, 0x00,
0x01, 0x00, 0x00, 0x00, 0x81, usleep(10000); 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00,
0x00, 0x00, 0x02, 0x00, 0x00, write(temp_fd,l_code,sizeof(l_code)); 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
0x96, 0x6f, 0x07, 0x00, 0x00, usleep(10000); 0x04, 0x00, 0x03, 0x00, 0x32, 0x00,
write(temp_fd,l_code,sizeof(l_code)); 0x02, 0x00, 0xa3, 0x85, 0x04, 0x00,
usleep(10000); 0x00, 0x00, 0x8f, 0x34, 0x02, 0x00,
write(temp_fd,o_code,sizeof(o_code)); 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

# Test result of Keyboard protectors (26/30)

- Keyboard protectors are neutralized
  - Most of the key values are revealed even if the arrangement of the key is randomized
  - Capture the screen when the key is pressed and compare the X,Y coordination



# Demonstration

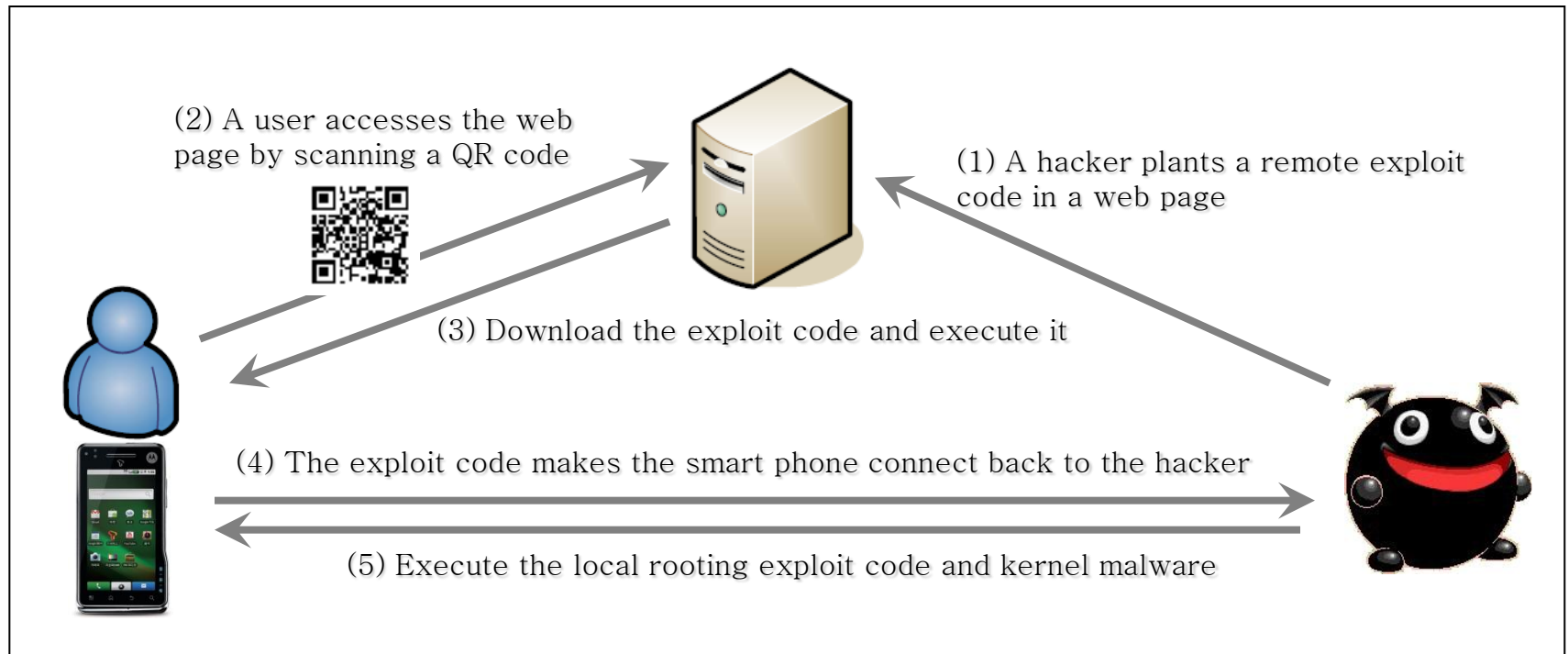
Android platform based kernel rootkit

- **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))



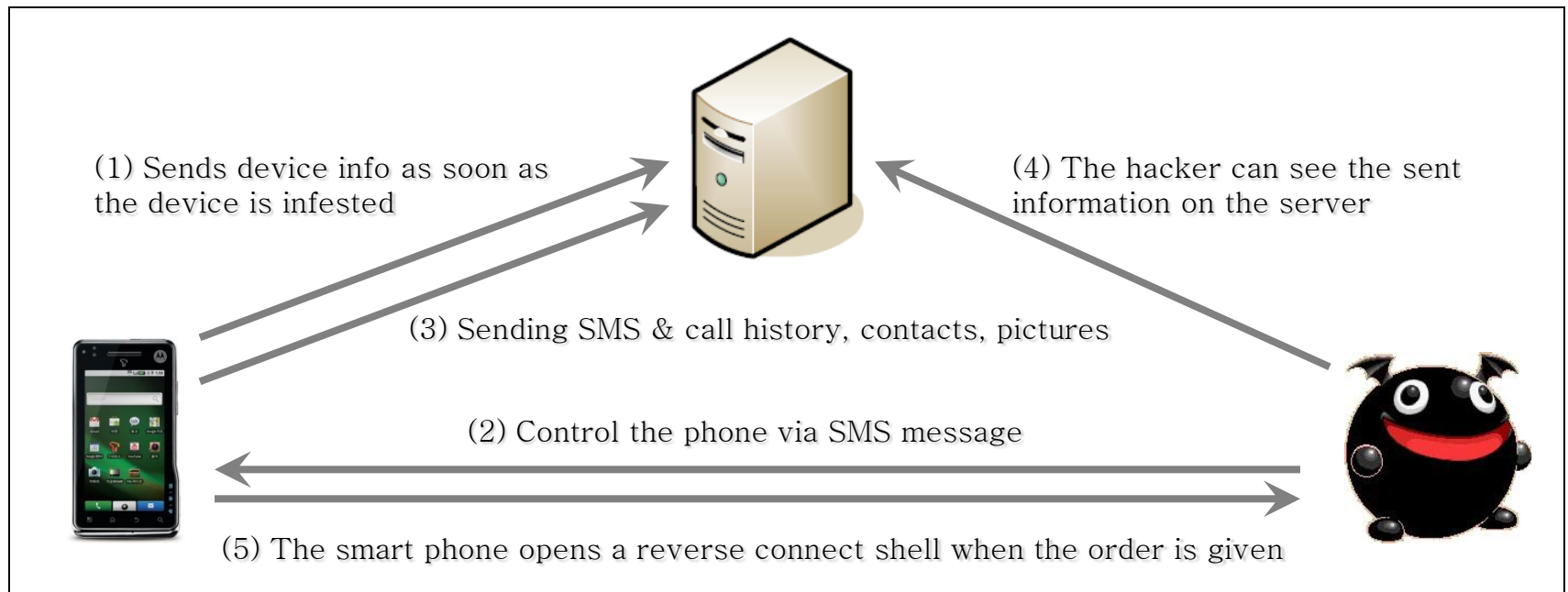
## ■ Android Remote / Local exploitation

- 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

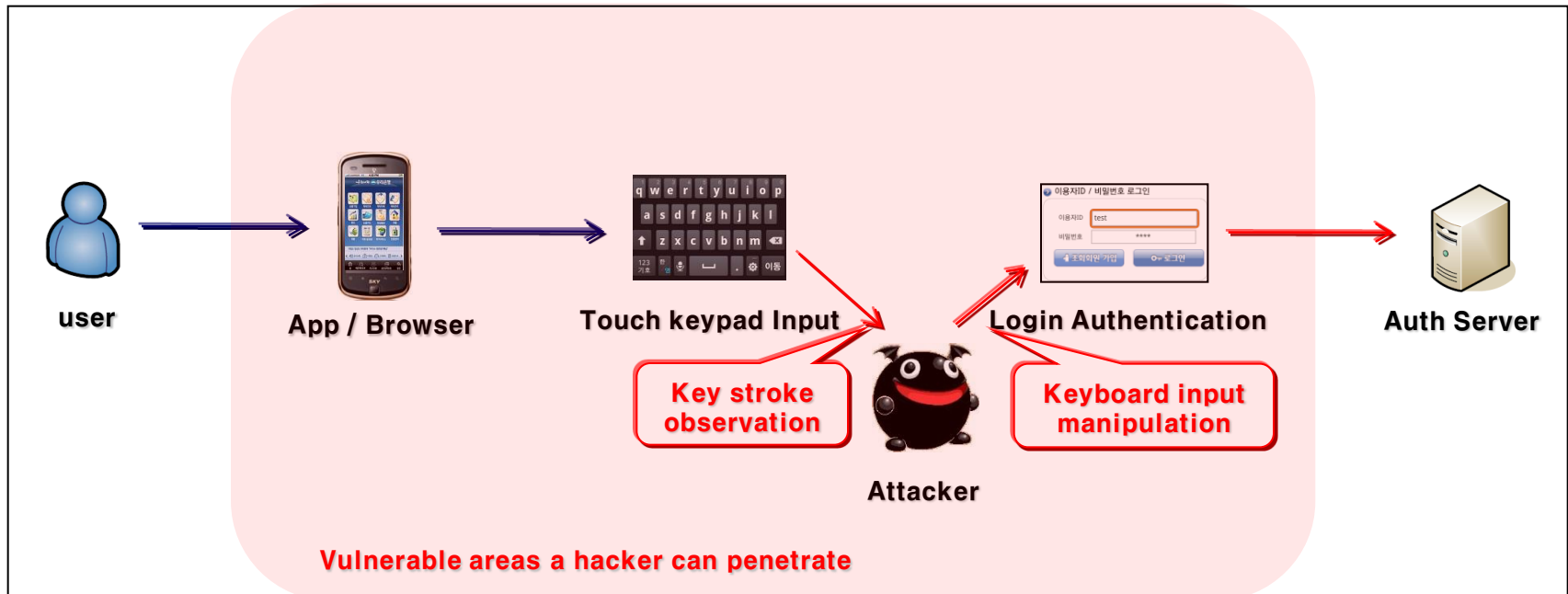


## ■ Android kernel rootkit

- 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



- **Demonstration of key logger**
  - Emulator/ Bluetooth keyboard logging
  - Touch pad key logging
  - Touch pad key injection
  - Neutralization of keyboard protector using screen capture



# Demonstration of Android Kernel Rootkit & Touchpad based Key logger



# Q & A



# Thank you !



By "dong-hoon yoU" (Xpl017Elz), in INetCop(c).  
MSN & E-mail: [szoahc\(at\)hotmail\(dot\)com](mailto:szoahc@hotmail.com)  
Home: <http://x82.inetcop.org>