HOW I FORCED AN ANDROID VULNERABILITY INTO BYPASSING MDM RESTRICTIONS + DIY MALWARE ANALYSIS

Zubair Ashraf
Team Lead & Security Researcher
IBM X-Force Advanced Research

@b0ut.m3

- Team Lead & Security Researcher @ IBM X-Force Research

@zashraf1337

securityintelligence.com/author/zubair-ashraf

ca.linkedin.com/in/zubairashraf

IBM X-Force® Research and Development

Expert analysis and data sharing on the global threat landscape

IP Reputation
Zero-day Research
URL / Web Filtering
Malware Analysis
Web Application Control
Vulnerability Protection
Anti-Spam
X FORCE

The IBM X-Force Mission

- **Monitor** and evaluate the rapidly changing threat landscape
- **Research** new attack techniques and develop protection for tomorrow's security challenges
- **Educate** our customers and the general public
- **Integrate** and distribute Threat Protection and Intelligence to make IBM solutions smarter

Agenda

- DIY Malware Analysis
- Vulnerability Hunt
- Exploitation

Android has Malware too ☺

Hon, my phone has been acting funny. You would know how to fix it, right?

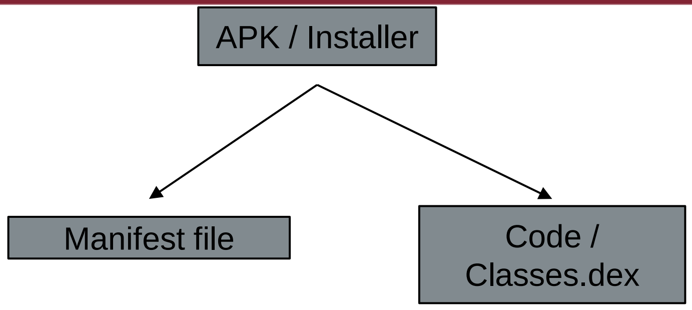
Jim, we got a sample of a highly sophisticated Malware, can you take a look?

Android Malware Analysis

The installer

- APK files are like zip / jar files
  - contains manifest file in binary format
  - use this to convert to human readable

```
java -jar AXMLPrinter2.jar AndroidManifest.xml
```



MIRcon 2014

Let’s give it a run

- Mobisec sourceforge.net/p/mobisec
- Notes on upgrading and installing additional tools bit.ly/UpgradMobiSec
- run it on top of your favorite virtualization product

MIRcon 2014

Try the free tools and services

- It's a good idea to test the free tools and services

- APKAnalyzer (apk-analyzer.net/) (dynamic)
- Dexter (dexter.dexlabs.org) - (static)

MIRcon 2014

Android Tools

▪SDK
developer.android.com/sdk/exploring.html
▪AVD
developer.android.com/tools/devices/managing-avds.htm
▪Emulator
developer.android.com/tools/help/emulator.html
▪ADB
developer.android.com/tools/help/adb.html

MIRcon 2014

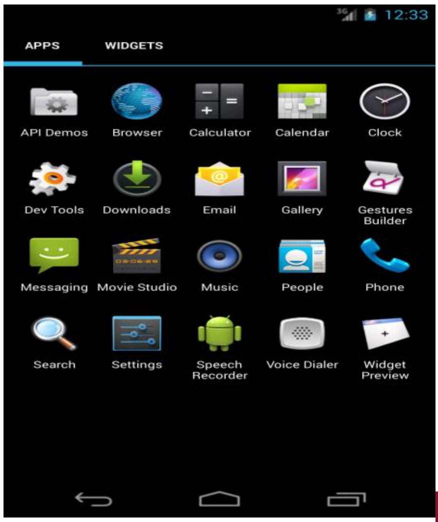
Let’s get the emulator running

```
mobisec@Mobisec:/opt/mobisec/devtools/android-sdk/tools$ emulator-arm -avd Android_4.0.3 -scale 0.75 -debug all -logcat all -no-boot-anim
```

```
mobisec@Mobisec-VM:~$ adb install Malware/OBad/E1064BFD836E4C895B569B2DE4700284.apk
```

MIRcon 2014

Find the app!



MIRcon 2014

Static Analysis of the apk

Package Name

- logcat entries / compare before & after output of adb shell pm list packages

drozer - mwrinfosecurity.com/products/drozer/

Static Analysis of the apk

drozer - mwrinfosecurity.com/products/drozer/

cd app.package
run info -a com.android.system.admin
run attacksurface com.android.system.admin
run manifest com.android.system.admin

# More interestingly:
run launchintent com.android.system.admin
tells us that the launcher activity for this package
com.android.system.admin.CCOIoll

Static Analysis of the apk

# Now if we wanted to manually launch this activity we can do so via:

```
dz#app.activity> run start --component
com.android.system.admin
com.android.system.admin.CCOIoll
```

MIRcon 2014

Static Analysis of the apk

# if we want to use the sdk tools only we can start this activity as:

```
mobisec@Mobisec:~$ adb shell am start -a \
android.intent.category.LAUNCHER -n
com.android.system.admin/.CCOIoll
```

MIRcon 2014

Nothing happens on screen even by launching manually

Dynamic Analysis via Debugger

Debugging

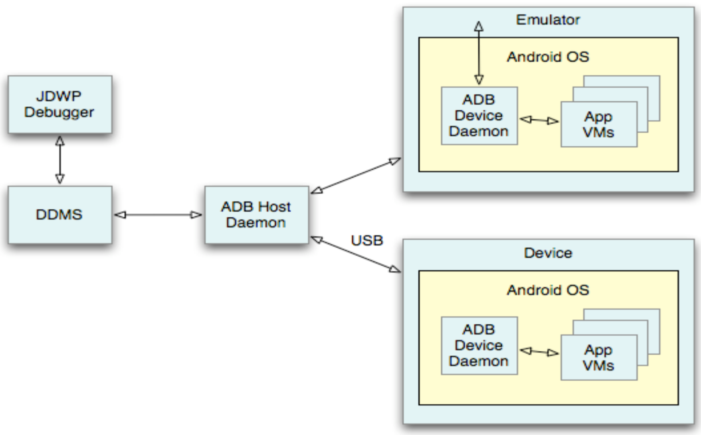


photo from http://developer.android.com/tools/debugging/index.html

Debugging

- On mobisec:
  /opt/mobisec/devtools/android-sdk/tools/monitor
- Emulator side:

  devtools ⇒ Development Settings ⇒ Debug app

  wait for debugger

Debugging

mobisec@Mobisec:~$ jdb -attach localhost:8700

- break on application entry point (using .jdbrc)
- stop in com.android.system.admin.COcCccl.onCreate
- trace go methods
  - gives you entry / exit log

MIRcon 2014

Debugging (caveat)

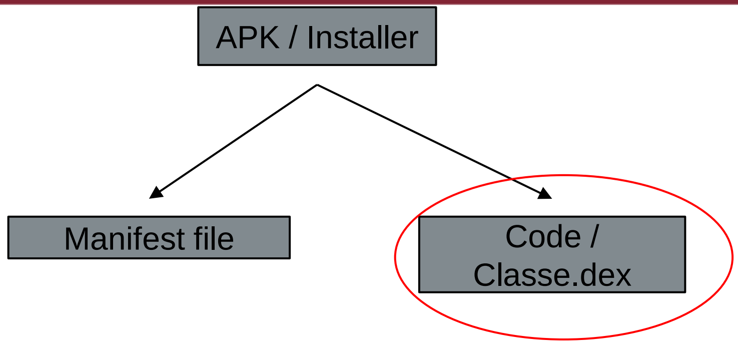
- last method entered
  - com.android.system.admin.COcCccl.onCreate
- app terminates
- Following are excluded from trace by default
  - “exclude” ⇐ configurable
    - java.*,javax.*,sun.*,com.sun.*

MIRcon 2014

Debugging

- exit method not seen because of “exclude”
- explicit breakpoint on
- stop in java.lang.System.exit(int)

Breakpoint hit: "thread=<1> main", java.lang.System.exit(),
[1] java.lang.System.exit (System.java:181), pc = 0
[2] com.android.system.admin.COcCccl.onCreate (null), pc = 1,041



Dex

Reference
http://www.strazzere.com/papers/DexEducation-PracticingSafeDex.pdf

Dex (Options …)

- Dex ⇒ java jar ⇒ java decompiler
- commercial decompiler like JEB
  - www.android-decompiler.com
- dex ⇒ IDA pro
- work with smali - code.google.com/p/smali
  - apktool - code.google.com/p/android-apktool/

Debugging at smali source level

- apktool version 2, supports this.
- java -jar apktool-cli-2.0.0-Beta5.jar d -d -o \
  decompiled_with_apktool_2_with_debug \
  d:\OBad\E1064BFD836E4C895B569B2DE4700284.apk

This will give you (among other things) java source files with smali code, e.g.

you will find COcCccl.java in decompiled_with_apktool_2_with_debug\smali\com\android\system\admin

Debugging at smali source level

code for onCreate you would see it as:

```
a=0;// # virtual methods
a=0;// .method public onCreate()V
a=0;// .locals 10
a=0;//
a=0;// invoke-super {p0}, Landroid/app/Application;-
>onCreate()V
a=0;//
a=0;// invoke-direct {p0}, Lcom/android/system/admin/COcCccl;-
>oIOccOcl()Z
```

Debugging at smali source level

code for onCreate you would see it as contd...

```
a=0;// move-result v0
a=0;//
a=0;// #v0=(Boolean);
a=0;// if-eqz v0, :cond_0
a=0;//
a=0;// const/4 v0, 0x1
a=0;//
a=0;// #v0=(One);
a=0;// invoke-static {v0}, Ljava/lang/System;->exit(I)V
```

Repackaging into an apk

- verify the manifest file
- aapt p --debug-mode -M \ d:\decompiled_with_apktool_2_with_debug\AndroidManifest.xml
- refer to specs to resolve errors - developer.android.com/guide/topics/manifest/manifest-intro.html

Repackaging into an apk contd …

D:\apktool_2\Apktool\brut.apktool\apktool-cli\build\libs>java -jar apktool-cli-2.0.0-Beta5.jar b -d -o E1064BFD836E4C895B569B2DE4700284_rebuilt_with_apktool_2_with_debug.apk d:\OBad\decompiled_with_apktool_2_with_debug

- signing your apk - you can read the details on android developer site, some reference commands below

Repackaging into an apk contd …

- creating keystore

D:\>"c:\Program Files\Java\jdk1.7.0_07\bin\keytool.exe" -genkeypair -validity 10000 -dname "CN=IBM-XF,C=CA" -keystore d:\downloads\MYKEYSTORE.keystore -storepass <keyPass> -keypass <Pass> -alias myXFKey -sigalg MD5withRSA -keyalg RSA -keysize 1024 -v

Repackaging into an apk contd …

- signing apk

D:\>"c:\Program Files\Java\jdk1.7.0_07\bin\jarsigner.exe" -keystore d:\downloads\MYKEYSTORE.keystore -storepass <keyPass> -keypass <Pass> -digestalg SHA1 -sigalg MD5withRSA -verbose -certs E1_rebuilt_apktool_2_dbg.apk myXFKey

Repackaging into an apk contd …

- zipalign - for optimization

D:\>zipalign -v 4 "d:\E1064BFD836E4C895B569B2DE4700284_rebuilt_with_apktool_2_with_debug.apk" "d:\E1_rebuilt_with_apktool_2_with_debug_aligned.apk"

- verifying jar signature -

D:\>"c:\Program Files\Java\jdk1.7.0_07\bin\jarsigner.exe" -verify -verbose -certs E1_rebuilt_apktool_2_dbg_aligned.apk

Debugging at smali source level

use /home/mobisec/Malware/OBAD/decompiled_with_apktool_2_with_debug/smali/

Breakpoint hit: "thread=<1> main", java.lang.System.exit(), line=181 bci=0

```
<1> main[1] wherei
[1] java.lang.System.exit (System.java:181), pc = 0
[2] com.android.system.admin.COcCccl.onCreate (COcCccl.java:5,758), pc = 1,041
```

MIRcon 2014

Debugging at smali source level (contd..)

```
# change frames, list source code, and examine variables
<1> main[1] up
<1> main[2] list
5,754 a=0;//
5,755 a=0;// const/4 v0, 0x0
5,756 a=0;//
5,757 a=0;// #v0=(Null);
5,758 => a=0;// invoke-static {v0}, Ljava/lang/System;->exit(I)V
5,759 a=0;//
5,760 a=0;// :cond_4
5,761 a=0;// #v0=(Boolean);
```

MIRcon 2014

Debugging at smali source level (contd..)

```
Lcom/android/system/admin/COcCccl;-
>oCIlCll:Landroid/content/Context;
5,763 a=0;//
<1> main[2] locals
Method arguments:
Local variables:
v9 = "dmBt"
v8 = instance of android.os.PowerManager(id=830019453032)
v2 = instance of byte[3] (id=830019585672)
v3 = "6311450ddea7b49349a92eeda1d528a5"
v1 = "sdk"
```

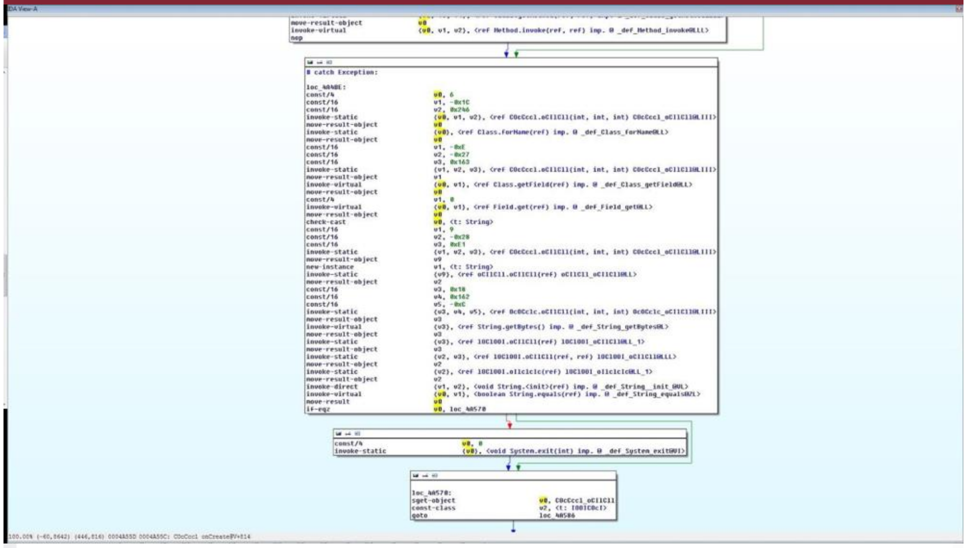
Debugging at smali source level (contd..)

```
a=0;// #v1=(Reference,Ljava/lang/String;);
a=0;// invoke-virtual {v0, v1}, Ljava/lang/String;-
>equals(Ljava/lang/Object;)Z //v0
a=0;// move-result v0
a=0;// #v0=(Boolean);
a=0;// if-eqz v0, :cond_4
a=0;//
a=0;// const/4 v0, 0x0
a=0;// #v0=(Null);
a=0;// invoke-static {v0}, Ljava/lang/System;->exit(I)V
```

Conditional jump?

IDA Pro

Can disassemble dex files



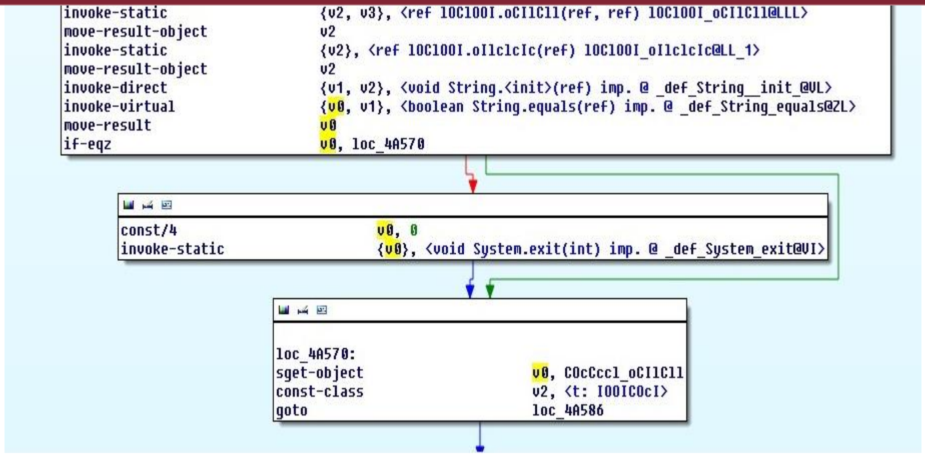
```
move-result-object         v0
invoke-virtual             {v0, v1, v2}, <ref Method.invoke(ref, ref) imp. @ _def_Method_invoke@LLL>
nop
```

```
# catch Exception:

loc_4A4BE:
const/4                    v0, 6
const/16                   v1, -0x1C
const/16                   v2, 0x246
invoke-static              {v0, v1, v2}, <ref C0cCccl.oCIlCIl(int, int, int) C0cCccl_oCIlCIl@LIII>
move-result-object         v0
invoke-static              {v0}, <ref Class.forName(ref) imp. @ _def_Class_forName@LL>
move-result-object         v0
const/16                   v1, -0xE
const/16                   v2, -0x27
const/16                   v3, 0x163
invoke-static              {v1, v2, v3}, <ref C0cCccl.oCIlCIl(int, int, int) C0cCccl_oCIlCIl@LIII>
move-result-object         v1
invoke-virtual             {v0, v1}, <ref Class.getField(ref) imp. @ _def_Class_getField@LL>
move-result-object         v0
const/4                    v1, 0
invoke-virtual             {v0, v1}, <ref Field.get(ref) imp. @ _def_Field_get@LL>
move-result-object         v0
check-cast                 v0, <t: String>
const/16                   v1, 9
const/16                   v2, -0x28
const/16                   v3, 0xE1
invoke-static              {v1, v2, v3}, <ref C0cCccl.oCIlCIl(int, int, int) C0cCccl_oCIlCIl@LIII>
move-result-object         v9
new-instance               v1, <t: String>
invoke-static              {v9}, <ref oCIlCIl.oCIlCIl(ref) oCIlCIl_oCIlCIl@LL>
move-result-object         v2
const/16                   v3, 0x18
const/16                   v4, 0x162
const/16                   v5, -0xC
invoke-static              {v3, v4, v5}, <ref 0c0Cclc.oCIlCIl(int, int, int) 0c0Cclc_oCIlCIl@LIII>
move-result-object         v3
invoke-virtual             {v3}, <ref String.getBytes() imp. @ _def_String_getBytes@L>
move-result-object         v3
invoke-static              {v3}, <ref 10C1001.oCIlCIl(ref) 10C1001_oCIlCIl@LL_1>
move-result-object         v3
invoke-static              {v2, v3}, <ref 10C1001.oCIlCIl(ref, ref) 10C1001_oCIlCIl@LLL>
move-result-object         v2
invoke-static              {v2}, <ref 10C1001.ollclclc(ref) 10C1001_ollclclc@LL_1>
move-result-object         v2
invoke-direct              {v1, v2}, <void String.<init>(ref) imp. @ _def_String__init_@VL>
invoke-virtual             {v0, v1}, <boolean String.equals(ref) imp. @ _def_String_equals@ZL>
move-result                v0
if-eqz                     v0, loc_4A570
```

```
const/4                    v0, 0
invoke-static              {v0}, <void System.exit(int) imp. @ _def_System_exit@VI>
```

```
loc_4A570:
sget-object                v0, C0cCccl_oCIlCIl
const-class                v2, <t: I00ICOcI>
goto                       loc_4A586
```

```
100.00% (-60,8642) (446,816) 0004A55D 0004A55C: C0cCccl onCreate@V+814
```

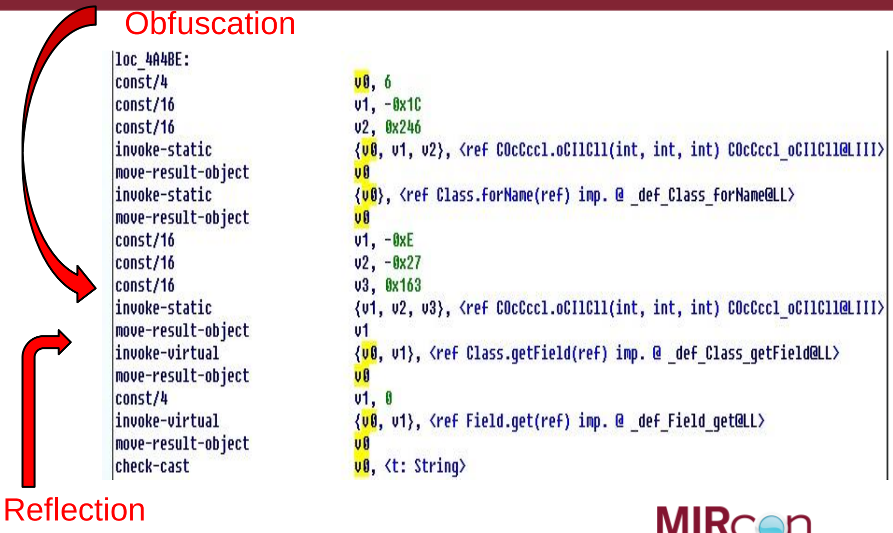


```
invoke-static          {v2, v3}, <ref 1OC1OOI.oCI1C11(ref, ref) 1OC1OOI_oCI1C11@LLL>
move-result-object     v2
invoke-static          {v2}, <ref 1OC1OOI.oIlclcIc(ref) 1OC1OOI_oIlclcIc@LL_1>
move-result-object     v2
invoke-direct          {v1, v2}, <void String.<init>(ref) imp. @ _def_String__init_@VL>
invoke-virtual         {v0, v1}, <boolean String.equals(ref) imp. @ _def_String_equals@ZL>
move-result            v0
if-eqz                 v0, loc_4A570

const/4                v0, 0
invoke-static          {v0}, <void System.exit(int) imp. @ _def_System_exit@VI>

loc_4A570:
sget-object            v0, COcCccl_oCI1C11
const-class            v2, <t: IOOICOcI>
goto                   loc_4A586
```

Exit if v0 equals v1, v1 comes from deobfuscating string

MIRcon 2014

v0 - deobfuscation + reflection



```
loc_4A4BE:
const/4                v0, 6
const/16               v1, -0x1C
const/16               v2, 0x246
invoke-static          {v0, v1, v2}, <ref COcCccl.oCI1C11(int, int, int) COcCccl_oCI1C11@LIII>
move-result-object     v0
invoke-static          {v0}, <ref Class.forName(ref) imp. @ _def_Class_forName@LL>
move-result-object     v0
const/16               v1, -0xE
const/16               v2, -0x27
const/16               v3, 0x163
invoke-static          {v1, v2, v3}, <ref COcCccl.oCI1C11(int, int, int) COcCccl_oCI1C11@LIII>
move-result-object     v1
invoke-virtual         {v0, v1}, <ref Class.getField(ref) imp. @ _def_Class_getField@LL>
move-result-object     v0
const/4                v1, 0
invoke-virtual         {v0, v1}, <ref Field.get(ref) imp. @ _def_Field_get@LL>
move-result-object     v0
check-cast             v0, <t: String>
```

MIRcon 2014

Finding all reflection calls

```
use <source path>
monitor print this
monitor locals
monitor where
monitor suspend
monitor cont
monitor resume
stop in java.lang.Class.getDeclaredField(java.lang.String)
stop in
java.lang.Class.getDeclaredMethod(java.lang.String,java.lang.Cl
ass[])
```

Finding all reflection calls

```
stop in java.lang.Class.getField(java.lang.String)
stop in
java.lang.reflect.AccessibleObject.setAccessible(boolean)
stop in java.lang.Runtime.exec(java.lang.String)
stop in java.lang.Runtime.exec(java.lang.String[])
stop in java.lang.System.exit(int)
```

- you can also try stopping in all forms of exec call

```
stop in java.lang.Runtime.exec(java.lang.String)
stop in java.lang.Runtime.exec(java.lang.String[])
```

Finding all reflection calls

```
# fields accessed
grep -E "name|this" \
OBAD_Reflection_Related_Code_Entries_Params_ST.txt

<1> main[1] this = "class
android.app.ActivityManager$RunningAppProcessInfo"
name = "processName"
...
this = "class android.os.Build"
name = "BRAND"
<1> main[1] this = "class android.os.Build"
name = "DEVICE"
```

Finding all reflection calls

```
# code places where os/dev specific fields were accessed
D:\>grep -E "name| \[1\]| \[2]"
OBAD_Reflection_Related_Code_Entries_Params_ST.txt

...
name = "MODEL"
[1] java.lang.Class.getField (Class.java:782)
[2] com.android.system.admin.COcCccl.onCreate
(COcCccl.java:5,683)
...
```

This was the reflection for
the exit we are investigating

Emulator Detection

various properties - adb shell get prop

compare the output with an actual device

Defeating Emulator Detection

- Modify the smali code
  fortiguard.com/sites/default/files/insomnidroid.pdf
  by @cryptax
- github.com/poliva/ldpreloadhook by @timstrazz
- Hack AOSP code

Hacking AOSP code

source.android.com/source/index.html

```
zashraf@ubuntu-10-x64:~/Android/src_4.3_r3$ diff
./frameworks/base/core/java/android/os/Build.java.modified
./frameworks/base/core/java/android/os/Build.java.orig
< import android.util.Log;
< public static String getString(String property) {
<     String p = SystemProperties.get(property, UNKNOWN);
<     Log.i("XF_IBM", "getString called for "+ property +" returning :" + p );
<     if (!property.equals("ro.product.model") && !p.equals("sdk"))
<       {
<       p = "Galaxy Nexus";
<       Log.i("XF_IBM", " Hooking return of SDK");
<       }
```

Hacking AOSP code

```
<     if (!property.equals("ro.product.name") && !p.equals("sdk"))
<       {
<       p = "yakju";
<       Log.i("XF_IBM", " Hooking return of SDK") ;
<       }
<       return p;
---
> private static String getString(String property) {
>     return SystemProperties.get(property, UNKNOWN);
```

Hacking AOSP code

target “aosp_arm-eng”

compile and obtained a fresh system.img file in

out/target/product/generic/

Hacking AOSP code

Creating a new AVD for emulator to run the custom built system.img

▪Create copies of android-18 in system-images and platforms sub directories under your sdk root direcotry. (I named the copies android-18_customized)+

▪Copy over the newly build system.img under the system-images folder (for my mobisec default config it was /opt/mobisec/devtools/android-sdk/system-images/android-18_customized/armeabi-v7)

Hacking AOSP code

* Make the following edits:
* in platform subdirectory

```
diff -r android-18/source.properties android-
18_customized/source.properties
8c8
< Platform.Version=4.3
---
> Platform.Version=4.3_Custom
14c14
< AndroidVersion.ApiLevel=18
---
> AndroidVersion.ApiLevel=18_custom
```

Hacking AOSP code

* Make the following edits too:

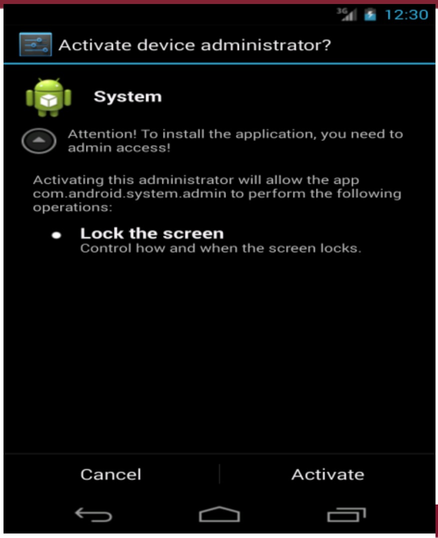
in system-images:

```
diff -r android-18/source.properties android-
18_customized/source.properties
8c8
< Platform.Version=4.3
---
> Platform.Version=4.3_Custom
14c14
< AndroidVersion.ApiLevel=18
---
> AndroidVersion.ApiLevel=18_custom
```

Hacking AOSP code

emulator-arm -avd Nexus_4_on_4.3_abi_18 -scale 0.75 -debug all -logcat all -no-boot-anim

The persistent begging starts



12:30

Activate device administrator?

System

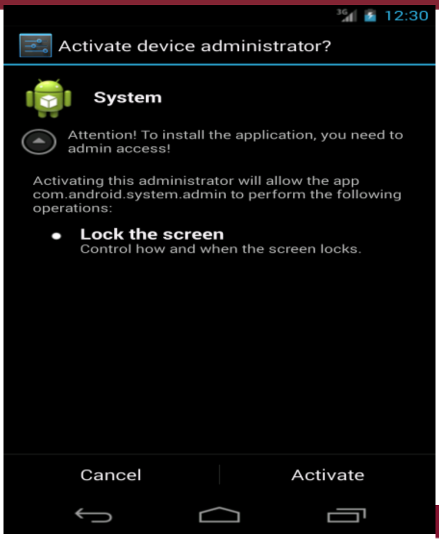
Attention! To install the application, you need to admin access!

Activating this administrator will allow the app com.android.system.admin to perform the following operations:

- **Lock the screen**
  Control how and when the screen locks.

Sales / Marketing

Won’t take No for an answer



12:30

Activate device administrator?

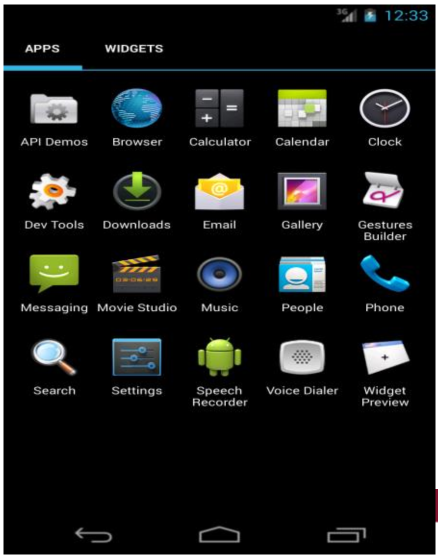
System

Attention! To install the application, you need to admin access!

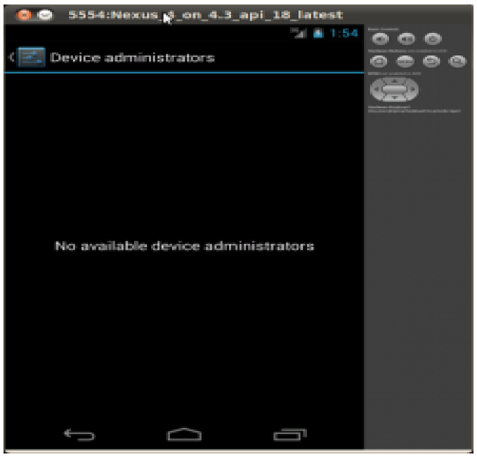
Activating this administrator will allow the app com.android.system.admin to perform the following operations:

- **Lock the screen**
  Control how and when the screen locks.

Let's launch the app now

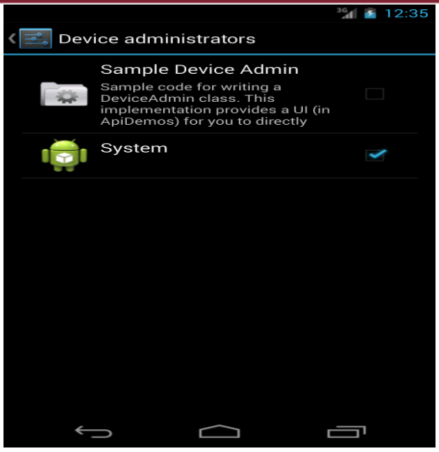


No launcher

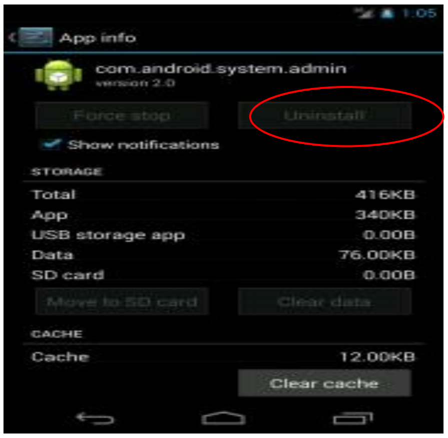


No available device administrators

No Device Admin?


12:35
Device administrators
Sample Device Admin
Sample code for writing a DeviceAdmin class. This implementation provides a UI (in ApiDemos) for you to directly
System

We would expect something like this


1:05
App info
com.android.system.admin
version 2.0
Force stop | Uninstall
Show notifications
STORAGE

Total	416KB
App	340KB
USB storage app	0.00B
Data	76.00KB
SD card	0.00B

Move to SD card | Clear data
CACHE

Cache	12.00KB

Clear cache

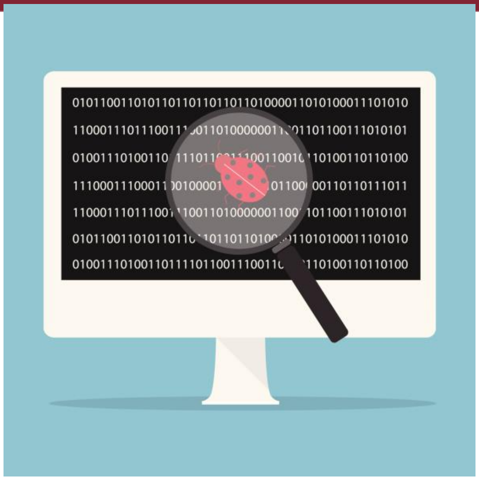
Can we see OBAD in app list and uninstall it?

Let’s try the command line

```
mobisec@Mobisec-VM:~/Malware/OBAD$ adb
uninstall com.android.system.admin
Failure

mobisec@Mobisec-VM:~/Malware/OBAD$ adb
logcat -d -b main -b events | grep admin | tail -1
W/PackageManager(  277): Not removing
package com.android.system.admin: has
active device admin
```

May be from command line - ‘adb’

Back to debugging



Hey! you are slowing us down

ACTIVITY NOT RESPONDING (ANR)

```
W/ActivityManager(  291): Timeout executing service:
ServiceRecord{421b03d8 u0
com.android.system.admin/.MainService}
...
W/ActivityManager(  291): Killing ProcessRecord{421b77a0
1129:com.android.system.admin/u0a10053}: background
ANR
```

Let’s Patch (1 of 2)

```
./frameworks/base/services/java/com/android/server/am/ActiveServices.java
<       if (!proc.debugging)
<          mAm.appNotResponding(proc, null, null, false, anrMessage);
<       else
<          Slog.w(TAG, "prevented ANR on debuggee app - Hijacked ANR
appnotresponding call for debugged app");
<
---
>       mAm.appNotResponding(proc, null, null, false, anrMessage);
```

Let’s Patch (1 of 2)

```
./frameworks/base/services/java/com/android/server/am/ActiveServices
.java
<       if (!proc.debugging)
<          mAm.appNotResponding(proc, null, null, false, anrMessage);
<       else
<          Slog.w(TAG, "prevented ANR on debuggee app - Hijacked ANR
appnotresponding call for debugged app");
<
---
>       mAm.appNotResponding(proc, null, null, false, anrMessage);
```

Let’s Patch (2 of 2)

~/Android/src_4.3_r3/frameworks/base/services/java/com/android/server/am/BroadcastQueue.java

```
<       if (!app.debugging)
<         mHandler.post(new AppNotResponding(app,
anrMessage));
<       else
<         Slog.w(TAG, "prevented ANR on (broadcast) debuggee
app - Hijacked ANR appnotresponding call for debugged app");
---
>       mHandler.post(new AppNotResponding(app,
anrMessage));
```

Tracing (incl Reflection) but avoiding other java code

```
exclude
android.os.*,org.apache.*,java.lang.D*,java.lang.N*,java.lang.P*,java.lang.U*,java.lang.F*,java.lang.Ru*,java.lang.E*,java.lang.T*,java.lang.V*,java.lang.I*,java.lang.A*,java.lang.S*,java.lang.B*,java.lang.ref.*,java.lang.C*,java.lang.O*,java.lang.S*,java.lang.V*,javax.*,sun.*,com.sun.*,java.s*,java.u*,java.s*,java.n*,java.i*,java.lang.reflect.A*,java.lang.reflect.C*,java.lang.reflect.F*,java.lang.reflect.Method.g*,java.lang.reflect.Method.<*
```

Launcher Hider

```
Breakpoint hit: "thread=<1> main",
android.app.ApplicationPackageManager.setComponentEnabledSetting

<1> main[1] wherei
  [1] android.app.ApplicationPackageManager.setComponentEnabledSetting (ApplicationPackageManager.java:1,262), pc = 0
…
  [3] java.lang.reflect.Method.invoke (Method.java:525), pc = 17
  [4] com.android.system.admin.cColOIOo.ollIIlc (null), pc = 748
```

MIRcon 2014

Launcher Hider

```
Breakpoint hit: "thread=<1> main",
<1> main[1] locals
Method arguments:
componentName = instance of
android.content.ComponentName(id=830033869864)
Local variables:
newState = 2
flags = 1

<1> main[1] print componentName
componentName =
"ComponentInfo{com.android.system.admin/com.android.system.admin.cColOIOo}"
```

MIRcon 2014

Let's hunt the code that hides it from Device Admin List

Checkout the patch history … or ...

Find Relevant Code

Launch *Settings -> Security -> Device Administrators*

Check out the logs:
adb logcat -d -b events

```
I/am_new_intent(  276):
[0,1106566944,17,com.android.settings/.Settings,android.intent.action.
MAIN,NULL,NULL,274726912]
I/am_resume_activity(  276):
[0,1106900904,17,com.android.settings/.Settings]
I/am_on_resume_called( 1118): [0,com.android.settings.Settings]
```

Find Relevant Code (contd…)

▪search for these strings at
androidxref.com

▪following along you will arrive at

packages/apps/Settings/src/com/android/settings/
DeviceAdminSettings.java

Find Relevant Code (contd…)

▪check out the function

void updateList()

▪and the conditions for something to appear in device admin list

Device Admin Vulnerability

getActivity().getPackageManager().queryBroadcastReceivers(Intent(**DeviceAdminReceiver.ACTION_DEVICE_ADMIN_ENABLED**), ...

Device Admin Vulnerability

getActivity().getPackageManager().queryBroadcastReceivers(Intent(**DeviceAdminReceiver.ACTION_DEVICE_ADMIN_ENABLED**), ...

Hackers won’t follow the specs unless they have to

What they should do

To use the Device Administration API, the application's manifest must include the following:

●A subclass of DeviceAdminReceiver that includes the following:

oThe BIND_DEVICE_ADMIN permission.

oThe ability to respond to the ACTION_DEVICE_ADMIN_ENABLED intent, expressed in the manifest as an intent filter.

What they actually did

```
<receiver "System" =".OCllCoO">
    <meta-data "android.app.device_admin"
="@2130968576">
    </meta-data>
    <intent-filter>
      <action
name="com.strain.admin.DEVICE_ADMIN_ENABLED">
      </action>
    </intent-filter>
  </receiver>
```

What they actually did

instead of

android.app.action.DEVICE_ADMIN_ENABLED

name="com.**strain**.admin.DEVICE_ADMIN_ENABLED">

What's next

Device Admin Vulnerability

services/java/com/android/server/

DevicePolicyManagerService.java

Device Admin Vulnerability

When adding an Admin

```
policy.mAdminMap.put(adminReceiver, newAdmin);
```

and

```
policy.mAdminList.add(newAdmin);
```

Please make sure you take ALL your stuff with you

Device Admin Vulnerability

removeActiveAdminLocked

```
1.policy.mAdminList.remove(admin);
2.policy.mAdminMap.remove(adminReceiver);
```

ALL THE TIME! even when in RUSH

Please make sure you take ALL your stuff with you

MIRcon 2014

Device Admin Vulnerability

```
private void handlePackagesChanged(int userHandle) {

removed = true;
policy.mAdminList.remove(i);
```

MIRcon 2014

Device Admin Vulnerability

```
private void handlePackagesChanged(int userHandle) {

removed = true;
policy.mAdminList.remove(i);
```

and who will clean up the
mAdminMap for you

Device Admin Vulnerability

This code path gets executed when you DISABLE the device admin component

Device Admin Vulnerability

All we have so far is a leak / bad coding practice

Device Admin Vulnerability

Is this a vulnerability?

Device Admin Vulnerability

Is there a code path that consults mAdminMap but not mAdminList ?

MIRcon 2014

Device Admin Vulnerability

- getActiveAdminUncheckedLocked
- getActiveAdminForCallerLocked
  (ComponentName who, int reqPolicy)
  with “who” parameter being non null

MIRcon 2014

Device Admin Vulnerability

getActiveAdminUncheckedLocked is used by isAdminActive

Device Admin Vulnerability

So can we exploit it?

Gartner:

DID YOU KNOW? By 2016, 20% of enterprise BYOD programs will fail due to deployment of **mobile device management (MDM)** measures that are too restrictive.

How about typing a 14 character password while driving?

Exploiting the Device Admin Vulnerability

▪enable device admin

▪disable the device admin component

▪At this point, from the data structure and code perspective, device admin's isAdminEnabled will still return true

Exploiting the Device Admin Vulnerability

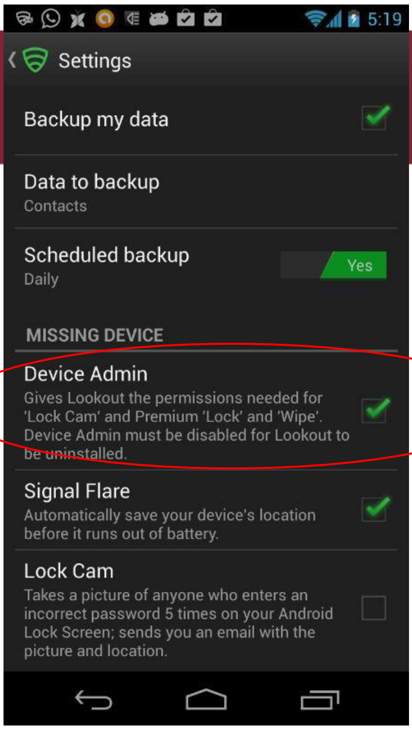
```
pm.setComponentEnabledSetting(
   this.getWho(context),
PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
   PackageManager.DONT_KILL_APP);
```

Exploiting the Device Admin Vulnerability

Uninstall the app (it will still be in the mAdminMap)

Exploiting the Device Admin Vulnerability

Now, install the original app



Settings

Backup my data ✓

Data to backup
Contacts

Scheduled backup
Daily
Yes

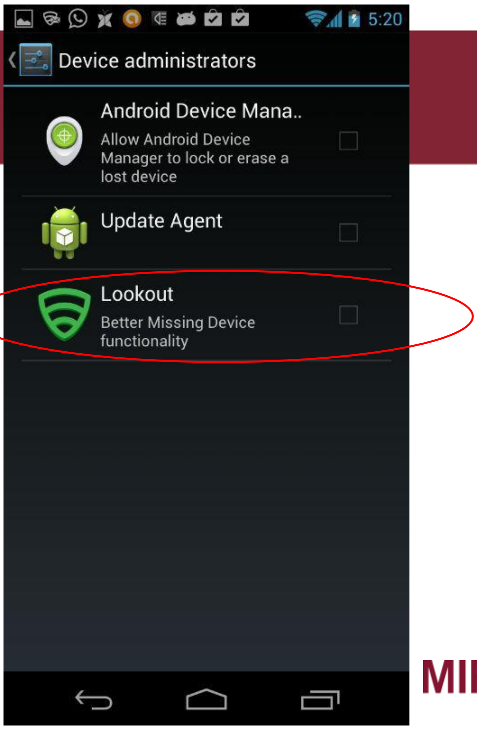
MISSING DEVICE

Device Admin ✓
Gives Lookout the permissions needed for 'Lock Cam' and Premium 'Lock' and 'Wipe'. Device Admin must be disabled for Lookout to be uninstalled.

Signal Flare ✓
Automatically save your device's location before it runs out of battery.

Lock Cam ☐
Takes a picture of anyone who enters an incorrect password 5 times on your Android Lock Screen; sends you an email with the picture and location.

BUT



Device administrators

Android Device Mana..
Allow Android Device Manager to lock or erase a lost device

Update Agent

Lookout
Better Missing Device functionality

BUT it may not necessarily work with MDM

isActivePasswordSufficient

isActivePasswordSufficient

public boolean isActivePasswordSufficient(int userHandle)
{
  enforceCrossUserPermission(userHandle);
  synchronized (this) {
    // This API can only be called by an active device
admin,
    DevicePolicyData policy = getUserData(userHandle);
    // so try to retrieve it to check that the caller is one.
    getActiveAdminForCallerLocked(null,
      DeviceAdminInfo.USES_POLICY_LIMIT_PASSWORD);

getActiveAdminForCallerLocked

```
ActiveAdmin getActiveAdminForCallerLocked
  (ComponentName who, int reqPolicy) throws
    SecurityException {
  if (who != null) { ... }
  else {
    final int N = policy.mAdminList.size();
```

getActiveAdminForCallerLocked

```
  else {
    final int N = policy.mAdminList.size();
    for (int i=0; i<N; i++) {
             ActiveAdmin admin = policy.mAdminList.get(i);
             if (admin.getUid() == callingUid &&
                 admin.info.usesPolicy(reqPolicy)) {
                return admin;
             }
         }
         throw new SecurityException
```

getActiveAdminForCallerLocked

```
else {
  final int N = policy.mAdminList.size();
  for (int i=0; i<N; i++) {
         ActiveAdmin admin = policy.mAdminList.get(i);
         if (admin.getUid() == callingUid &&
            admin.info.usesPolicy(reqPolicy)) {
            return admin;
         }
      }
      throw new SecurityException
```

There is a way

sharedUID

▪active device admin with same policies

▪and same UID - sharedUID

```
if (admin.getUid() == callingUid &&
    admin.info.usesPolicy(reqPolicy)) {
```

Extended Hack

▪Modify AndroidManifest.xml of the MDM

−add android:sharedUserId attribute

▪repackage and self sign

Extended Hack

▪Create a different device admin

−same sharedUid

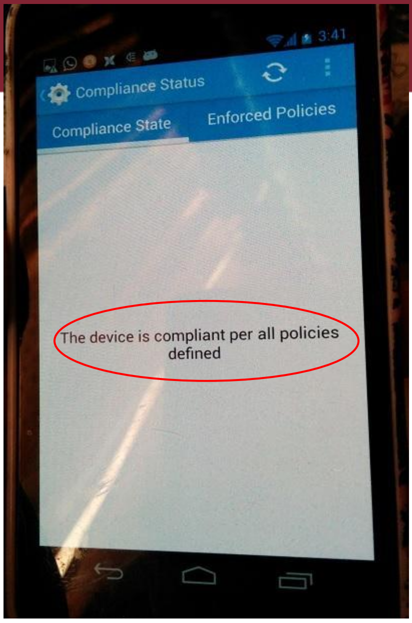
−same policies

−install and activate it

Extended Hack

▪Do everything else as before

−but using the self signed MDM apk with sharedUID



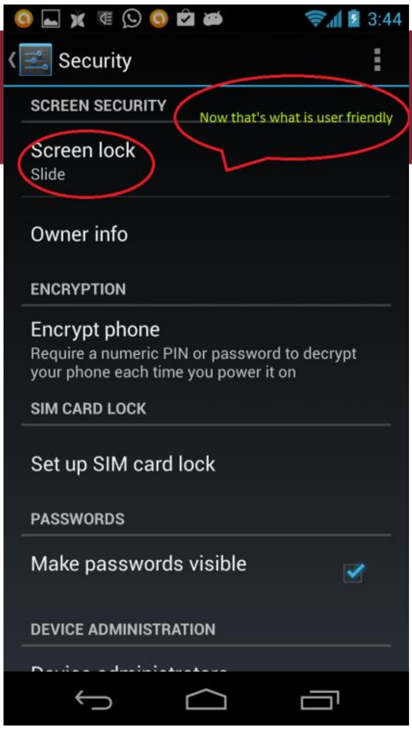
Compliance Status

Compliance State

Enforced Policies

The device is compliant per all policies defined

COMPLIANT != SECURE


Security

SCREEN SECURITY

Screen lock
Slide

Owner info

ENCRYPTION

Encrypt phone
Require a numeric PIN or password to decrypt your phone each time you power it on

SIM CARD LOCK

Set up SIM card lock

PASSWORDS

Make passwords visible ✓

DEVICE ADMINISTRATION

Lessons

- Don’t make it really painful to use the device
- code protection
- verifying app signatures

Further Learning

- https://github.com/strazzere/android-unpacker
- https://github.com/strazzere/android-unpacker/blob/master/AHPL0.pdf

First to The Creator

Loved ones, X-Force & MIRCon and YOU

@zashraf1337

securityintelligence.com/author/zubair-ashraf

ca.linkedin.com/in/zubairashraf