Dynamic Analysis using CobraDroid

PRESENTED BY: Jake Valletta

OCTOBER 18, 2013

About Me

- Consultant at Mandiant
- Pen-testing, IR, forensics, application security
  - Strong interests in mobile security
- Mobile security blog and research: “The Cobra Den”
  - http://blog.thecobraden.com/
  - http://www.thecobraden.com/
- @jake_valletta

Agenda

- Background & Overview
- CobraDroid Features
- Demo
- Future Plans
- Questions & Answers

Background & Overview

Current Situation – Background

- People want/need to analyze Android applications
  - Companies pay to be told they are “safe”
  - Analyzing malware
  - General curiosity (why is Angry Birds asking to use my camera?)

Current Situation – Static Analysis?

- Lots of tools!
  - Smali/Baksmali
  - Dex2jar
  - Apktool
  - Dexter by BlueBox
  - IDA Pro
- Lots of information on how to tear applications apart…
  - …And modify and repackage!

Current Situation – Dynamic Analysis?

- There are plenty of services that will analyze your application
  - Upload to website, get results
  - NOT ideal for client related work
  - “Blackbox” approach
- Stand-alone solutions less common
  - “AppUse” by AppSecLabs (closed-source)

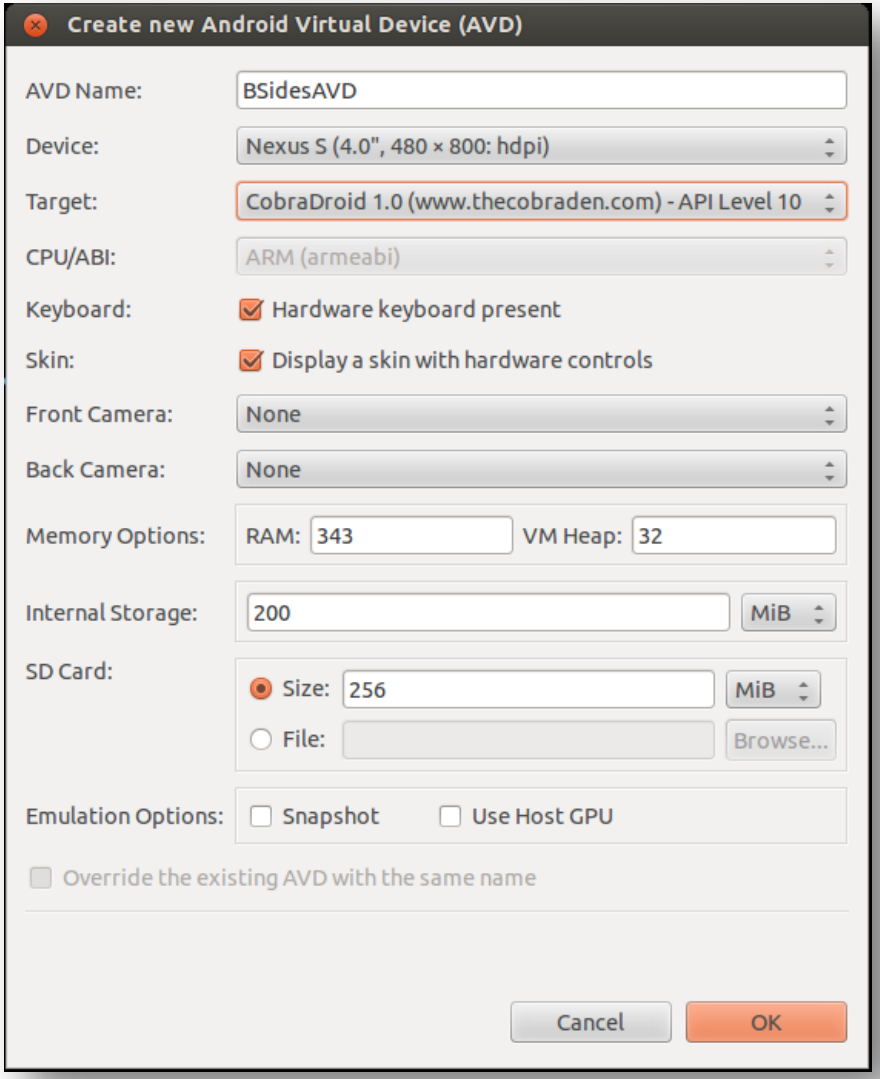
Goals of CobraDroid

- Create a free and open dynamic analysis platform
  - Needs to be easy to install, setup, and use
- Give the tester as much control and visibility as possible
  - Make their job easier and successful

What is CobraDroid?

- Modified Android build for the emulator
  - QEMU emulating ARM code
  - Android 2.3.7 (GingerBread)
- Modified from lowest point up
  - Kernel
  - User-space libraries & tools
  - Dalvik virtual machine (VM)
  - Android applications

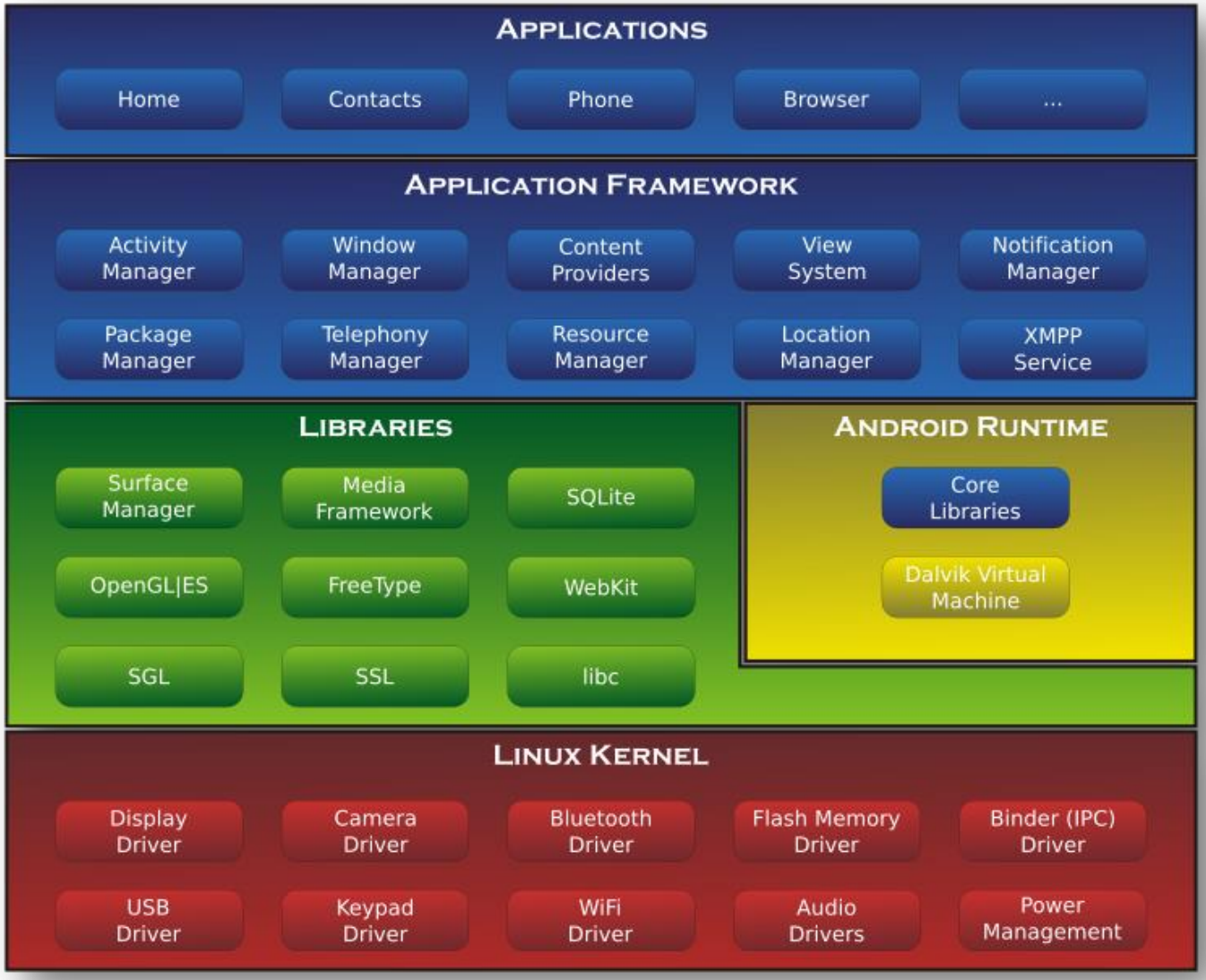
Using CobraDroid



1. Setup Android SDK
2. Download CobraDroid archive from my website
3. Unzip to “add-ons” directory (SDK)
4. Create new AVD, “Target” CobraDroid 1.0

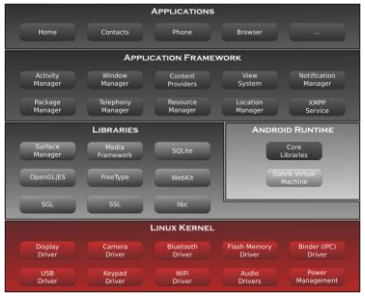
CobraDroid Features

Android Architecture



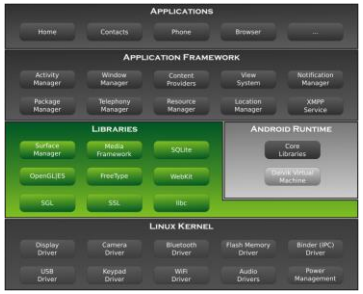
Updated Kernel

- At the time of development, latest “Goldfish” kernel was 2.6.29
  - “kernel.org” publish date of April 13, 2008
  - Default kernel with Android 1.5 “Donut” (released Sept 19, 2009)
- Updated to 2.6.36
  - Default kernel with Android 3.0 “HoneyComb” (released Feb 22, 2011)
- More powerful configuration
  - Full netfilters (ProxyDroid, iptables)
  - Loadable kernel modules



Bash & BusyBox

- Android 2.3.x shell is terrible. ***Terrible***.
  - No autocomplete
  - No coloring
  - No pipes
- Lack of tools/utilities
  - No editors
  - No ${your_favorite_tool}



Bash & BusyBox

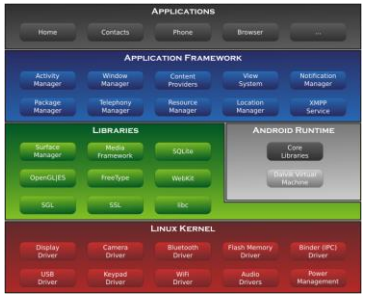
```
root@android-assessment:/home/analyst# adb shell
CobraDroid / # uname -a
Linux localhost 2.6.36-CobraKernel #102 Mon Jul 29 13:38:48 EDT 2013 armv5tejl GNU/Linux
CobraDroid / # ls -la /
drwxr-xr-x   13 root     root            0 Sep 17 19:55 .
drwxr-xr-x   13 root     root            0 Sep 17 19:55 ..
drwxr-xr-x    3 root     root            0 Sep 17 19:55 acct
drwxrwx---    1 system   cache        2048 Sep 17 19:55 cache
dr-x------    2 root     root            0 Sep 17 19:55 config
lrwxrwxrwx    1 root     root           17 Sep 17 19:55 d -> /sys/kernel/debug
drwxrwx--x    1 system   system       2048 Sep 17 19:56 data
-rw-r--r--    1 root     root          118 Dec 31  1969 default.prop
drwxr-xr-x   10 root     root         2120 Sep 17 19:56 dev
lrwxrwxrwx    1 root     root           11 Sep 17 19:55 etc -> /system/etc
-rwxr-x---    1 root     root        94168 Dec 31  1969 init
-rwxr-x---    1 root     root         1731 Dec 31  1969 init.goldfish.rc
-rwxr-x---    1 root     root        13827 Dec 31  1969 init.rc
drwxrwxr-x    6 root     system          0 Sep 17 19:55 mnt
dr-xr-xr-x   77 root     root            0 Dec 31  1969 proc
drwx------    2 root     root            0 Jul 23 18:55 root
drwxr-x---    2 root     root            0 Dec 31  1969 sbin
lrwxrwxrwx    1 root     root           11 Sep 17 19:55 sdcard -> /mnt/sdcard
drwxr-xr-x   12 root     root            0 Sep 17 19:55 sys
drwxr-xr-x    1 root     root         2048 Sep 15 17:44 system
-rw-r--r--    1 root     root            0 Dec 31  1969 ueventd.goldfish.rc
-rw-r--r--    1 root     root         3882 Dec 31  1969 ueventd.rc
lrwxrwxrwx    1 root     root           14 Sep 17 19:55 vendor -> /system/vendor
CobraDroid / #
```

Bash & BusyBox

```
root@android-assessment:/home/analyst# adb shell
CobraDroid / # uname -a
Linux localhost 2.6.36-CobraKernel #102 Mon Jul 29 13:38:48 EDT 2013 armv5tejl GNU/Linux
CobraDroid / # ls -la /
drwxr-xr-x   13 root     root            0 Sep 17 19:55 .
drwxr-xr-x   13 root     root            0 Sep 17 19:55 ..
drwxr-xr-x    3 root     root            0 Sep 17 19:55 acct
drwxrwx---    1 system   cache        2048 Sep 17 19:55 cache
dr-x------    2 root     root            0 Sep 17 19:55 config
lrwxrwxrwx    1 root     root           17 Sep 17 19:55 d -> /sys/kernel/debug
drwxrwx--x    1 system   system       2048 Sep 17 19:56 data
-rw-r--r--    1 root     root          118 Dec 31  1969 default.prop
drwxr-xr-x   10 root     root         2120 Sep 17 19:56 dev
lrwxrwxrwx    1 root     root           11 Sep 17 19:55 etc -> /system/etc
-rwxr-x---    1 root     root        94168 Dec 31  1969 init
-rwxr-x---    1 root     root         1731 Dec 31  1969 init.goldfish.rc
-rwxr-x---    1 root     root        13827 Dec 31  1969 init.rc
drwxrwxr-x    6 root     system          0 Sep 17 19:55 mnt
dr-xr-xr-x   77 root     root            0 Dec 31  1969 proc
drwx------    2 root     root            0 Jul 23 18:55 root
drwxr-x---    2 root     root            0 Dec 31  1969 sbin
lrwxrwxrwx    1 root     root           11 Sep 17 19:55 sdca
drwxr-xr-x   12 root     root            0 Sep 17 19:55 sys
drwxr-xr-x    1 root     root         2048 Sep 15 17:44 syst
-rw-r--r--    1 root     root            0 Dec 31  1969 uever
-rw-r--r--    1 root     root         3882 Dec 31  1969 ueven
lrwxrwxrwx    1 root     root           14 Sep 17 19:55 vendor
CobraDroid / #
```

LiME Forensics

- Linux Memory Extractor by Joe Sylve (504ensics)
  - http://code.google.com/p/lime-forensics/
- Allows for live memory acquisition via Loadable Kernel Module
  - Open saved files with Volatility or Dalvik Inspector
- Modified to fit CobraDroid as device driver + user-space API
  - https://github.com/jakev/lime-forensics-jakev



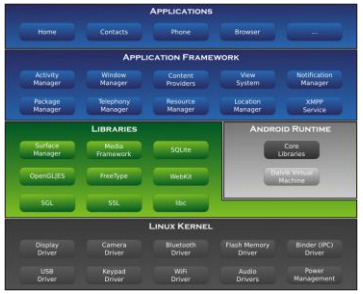
LiME Forensics

- “lime” command line utility
  - Links against “liblime.so”
- “android.jakev.Lime” class for Android applications
  - **NOT SAFE!** Currently implementing safer solution
  - Gives Android application access to kernel driver

```
CobraDroid / # lime -d/mnt/sdcard/memory.dump -fraw
Disk mode selected: /mnt/sdcard/memory.dump
Output format: raw
About to dump memory to disk...
```

Editable Radio & Device Identifiers

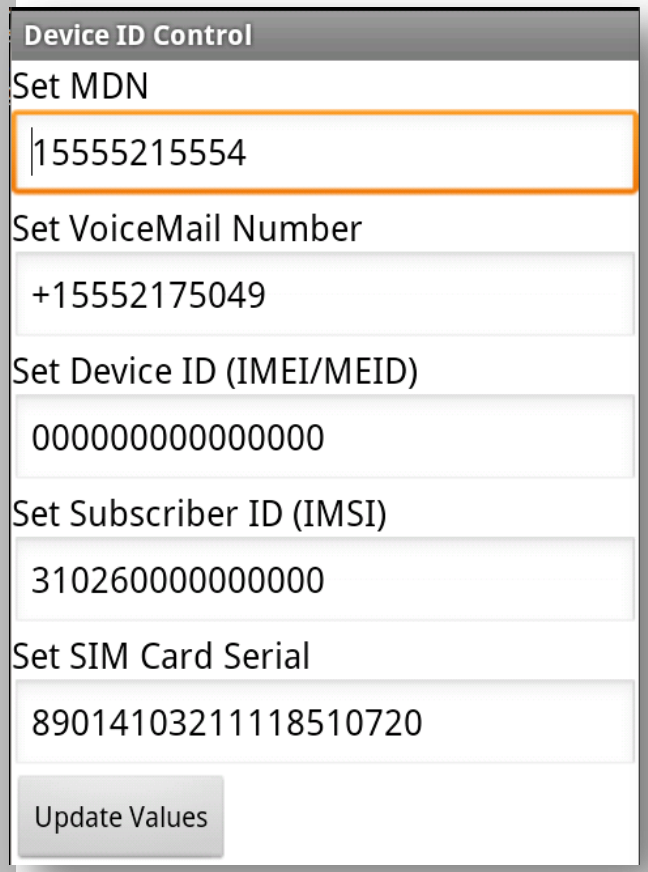
- Lets you make the phone look like anything you want!
- Helps with application whitelisting/blacklisting
  - Is the carrier Verizon, or AT&T? Is it a Nokia? Motorola?
- Previously very tedious to change on emulator
  - Radio properties: Modify “emulator-arm” binary
  - Device properties: Modify :“/etc/build.prop” and reconstruct the “system.img”

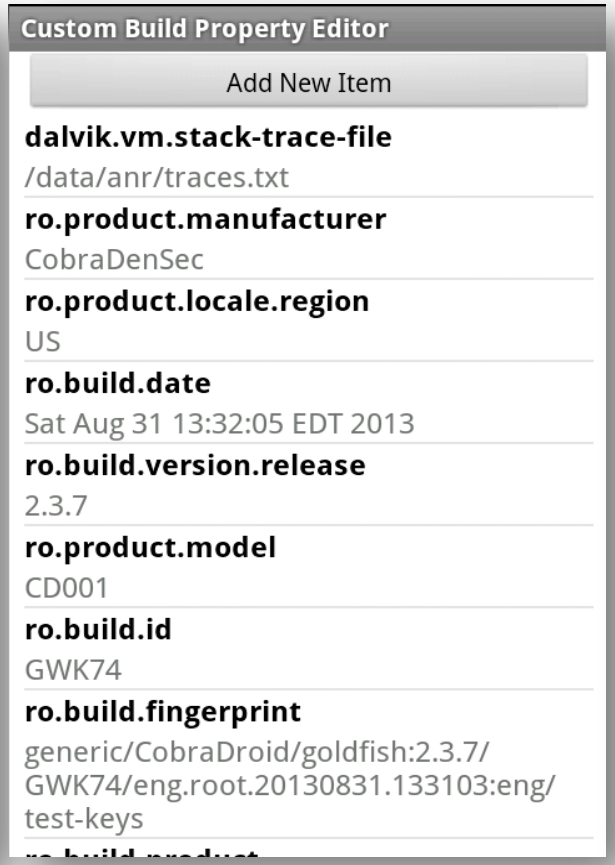


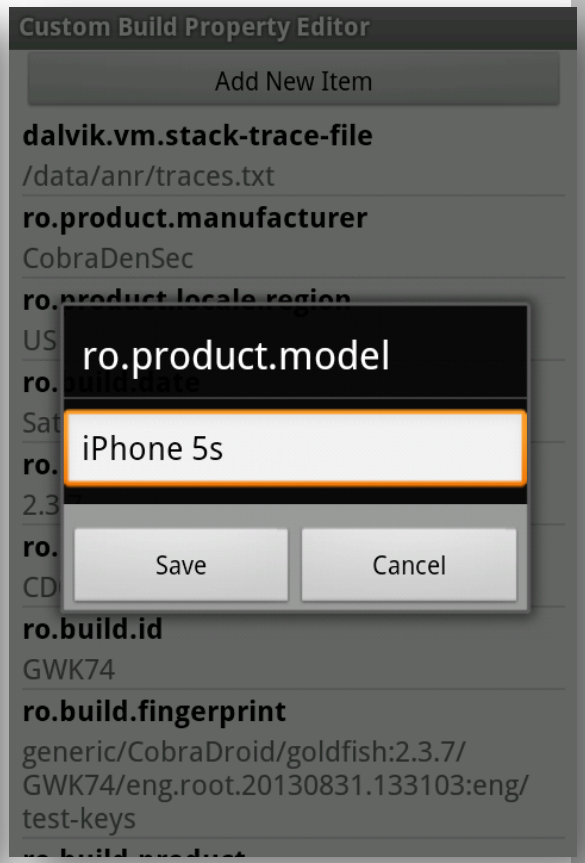
Editable Radio & Device Identifiers

- Re-written “TelephonyManager” class
  - Queries a custom file instead
- Removed “android.os.Build” class initialization in Zygote
  - Hooked “SystemProperties” class
  - Queries a custom file instead

Editable Radio & Device Identifiers

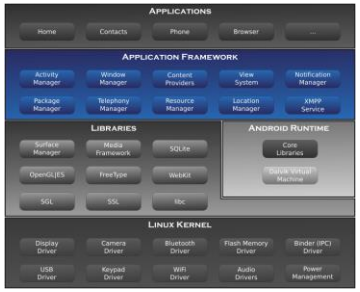






SSL Validation Bypass

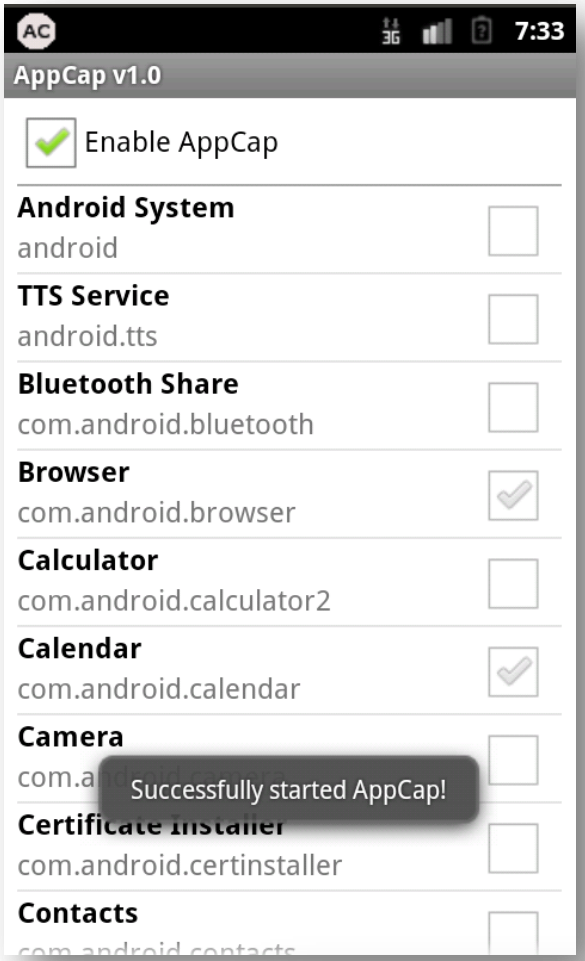
- Allows you to man-in-the-middle any SSL connection
  - Disables certificate pinning and CA validation silently
- Re-written constructors and getter/setters
- Works for all default SSL libraries on Android 2.3
  - HttpsURLConnection (core.jar)
  - DefaultHttpClient (ext.jar)
  - SSLSocketFactory (ext.jar)

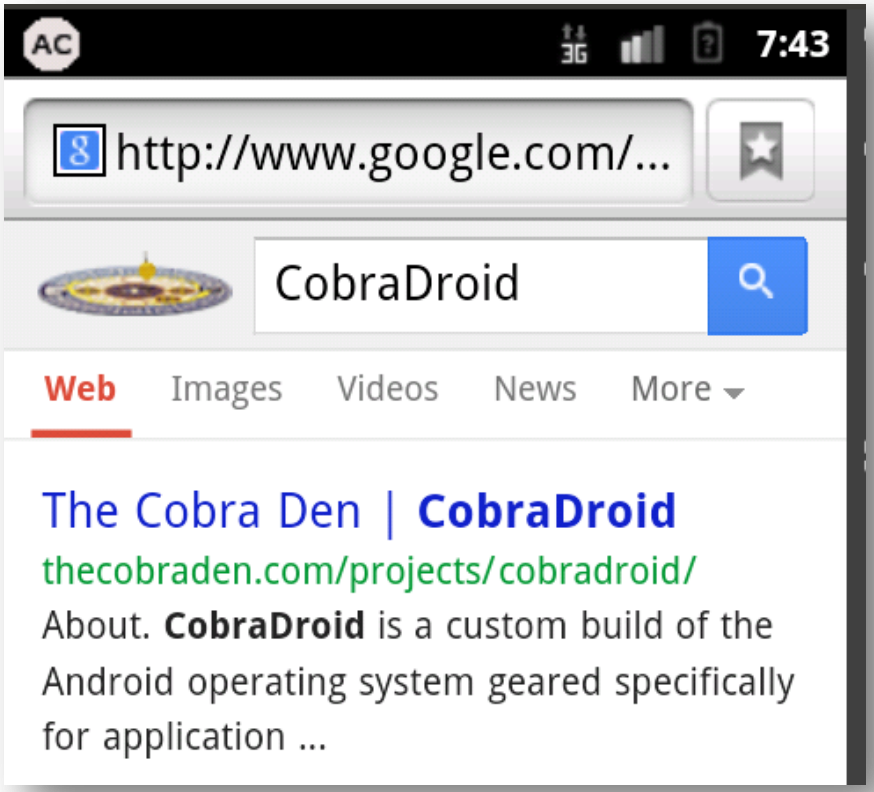


Application Specific Packet Capture

- *Show me only traffic for application X (and application Y)*
  - Focus on only the traffic you actually care about
- Uses Custom “iptables” rules to redirect traffic
- View in Wireshark afterwards
  - Tested on 1.8.5 Stable, 1.11.0 Dev. (incompatible with older versions)

Application Specific Packet Capture





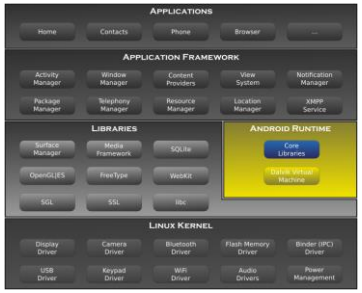

```
CobraDroid /mnt/sdcard # ls -l
d---rwxr-x    2 system   sdcard_r      2048 Sep 18 18:32 LOST.DIR
----rwxr-x    1 system   sdcard_r    641024 Sep 18 19:45 appcap-20130918_193754.pcap
CobraDroid /mnt/sdcard #
```

Android Packages

- ProxyDroid
  - Makes it painless to proxy traffic on the emulator
- Superuser/“su”
  - Provides root level access to the device
- Drozer
  - Allows you to assume the role of an Android application at a command line
- EmuCoreTools
  - Front-end interface to CobraDroid features

Method Call Alerting

- Hooked Dalvik VM to alert when a method is called
  - Lots of potential here!
- Could have an entire 45 minute talk on hooking the DVM
  - I'm going to try and do it in about 7 ☺
- **TL;DR – Instrumenting method byte-code during Class loading**



Method Call Alerting

```
#Prototype Hook Configuration File
# v0.1

# Our System Hook Section
.sys
# Getting radio parameteres
android.telephony.TelephonyManager
        getDeviceId @Alert
        getLine1Number @Alert
        getSubscriberId @Alert

# Getting environment directories
android.os.Environment
        getExternalStorageDirectory @Alert

# Sending SMS
android.telephony.gsm.SmsManager
        sendDataMessage @Alert "Send a data based SMS to specific application port"
        sendMultipartTextMessage @Alert
        sendTextMessage @Alert

.end

#Our Application Hook Section
.app
# An Application Hook
com.jakev.testing.TestingActivity
         nzkds @Alert "Obfuscated method is accessing your contacts!"
.end
```

Method Call Alerting

```
#Prototype Hook Configuration File
# v0.1

# Our System Hook Section
.sys
# Getting radio parameteres
android.telephony.TelephonyManager
        getDeviceId @Alert
        getLine1Number @Alert
        getSubscriberId @Alert

# Getting environment directories
android.os.Environment
        getExternalStorageDirectory @Alert

# Sending SMS
android.telephony.gsm.SmsManager
        sendDataMessage @Alert "Send a data based SMS to specific application port"
        sendMultipartTextMessage @Alert
        sendTextMessage @Alert

.end

#Our Application Hook Section
.app
# An Application Hook
com.jakev.testing.TestingActivity
         nzkds @Alert "Obfuscated method is accessing your contacts!"
.end
```

System Hooks

Application Hooks

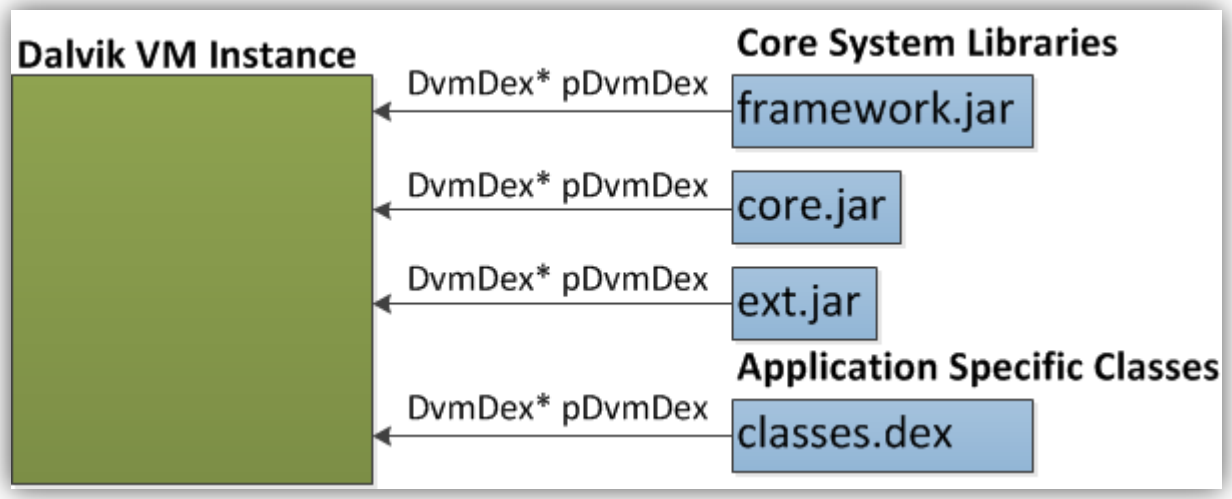
Method Call Alerting

- It's magic! (…right?)

```
root@android-assessment:/home/analyst# adb logcat -b security
D/EventNotifier(  575): [com.jakev.testing] An application accessed your device ID! "android.telephony.TelephonyManager.getDeviceId()"
D/EventNotifier(  575): [com.jakev.testing] Method Call Alert: "android.os.Environment.getExternalStorageDirectory()"
D/EventNotifier(  575): [com.jakev.testing] Method Call Alert: "com.jakev.testing.TestingActivity.snakeTestCall()"
D/EventNotifier(  575): [com.jakev.testing] Obfuscated method is accessing your contacts! "com.jakev.testing.TestingActivity.nzkds()"
^C
root@android-assessment:/home/analyst#
```

Step #1 – DVM Startup

- Read configuration file and parse hooks into global DVM memory
  - Utilize the “gDvm” variable (DvmGlobals struct)
- Allocate additional space for new data based on configuration
  - Modify calloc() calls when initializing “pDvmDex” (DvmDex struct)



Step #2 – Class/Method Loading

- Read global memory to determine if loaded class and method should be hooked
- If we hit a hooked method, we need to allocate additional space for new instructions (DexCode struct)
  - The original DexCode structure is read-only mapped directly from the DEX file…
  - We allocate a new DexCode structure, and use that instead!

“DexCode” Structure

Name	Format
registers_size	u2
ins_size	u2
outs_size	u2
tries_size	u2
debug_info_off	u4
insns_size	u4
insns	u2[insns_size]
padding	u2
tries	try_item[tries_size]
handlers	encoded_catch_handler_list

- Contains all details for a method

“DexCode” Structure

Name	Format
registers_size	u2
ins_size	u2
outs_size	u2
tries_size	u2
debug_info_off	u4
insns_size	u4
insns	u2[insns_size
padding	u2
tries	try_item[tries
handlers	encoded_cat

- Contains all details for a method
- Add new instructions to “insns”
- Repair DexCode structure pointers

```
                        |[2a1fdc] android.telephony.Telepl
1202                    |0000: const/4 v2, #int 0 // #0
7010 c655 0300          |0001: invoke-direct {v3}, Landroi
0c01                    |0004: move-result-object v1
7210 5d8b 0100          |0005: invoke-interface {v1}, Lcom
0c01                    |0008: move-result-object v1
1101                    |0009: return-object v1
0d01                    |000a: move-exception v1
0710                    |000b: move-object v0, v1
0721                    |000c: move-object v1, v2
28fc                    |000d: goto 0009 // -0004
0d01                    |000e: move-exception v1
```

Step #2 – Class/Method Loading

- Add instructions to beginning of “insns” to call the “EventNotifier” class
  - “notifyEvent” is responsible for printing to the logs
  - **Assumes we know the location of this class/method**

```
LOGI("There is no payload. Adding 3 insns.");
/* invoke-static {}, Landroid/jakev/EventNotifier;.notifyEvent
 * [71 35c]
 * B|A|op CCCC G|F|E|D
 * [B=0] op {}, kind@CCCC
 * A=0, B=0 op=71, GEFD=0
 * 00 71 [CC CC] 00 00
 */
pDexCode->insns[0] = 0x0071;
pDexCode->insns[1] = eventNotifierMethId;
pDexCode->insns[2] = 0x0000;
```

Step #3 – Resolving

- Resolving occurs at runtime, when the DVM must determine what code to run and where it is located

```
private void test() {
    String s1 = "HERE";
    String s2 = "BSides Rulez";
    Log.d(s1, s2);
}
```

Step #3 – Resolving

- Resolving occurs at runtime, when the DVM must determine what code to run and where it is located

```
private void test() {
    String s1 = "HERE";
    String s2 = "BSides Rulez";
    Log.d(s1, s2);
}
```

```
                |[0009d8] com.jakev.testing.TestingActivity.test:()V
1a00 0700       |0000: const-string v0, "HERE" // string@0007
1a01 0100       |0002: const-string v1, "BSides Rulez" // string@0001
7120 0700 1000  |0004: invoke-static {v0, v1}, Landroid/util/Log;
                                .d:(Ljava/lang/String;Ljava/lang/String;)I // method@0007
0e00            |0007: return-void
```

Step #3 – Resolving

- Resolving occurs at runtime, when the DVM must determine what code to run and where it is located

```
private void test() {
    String s1 = "HERE";
    String s2 = "BSides Rulez";
    Log.d(s1, s2);
}
```

```
                |[0009d8] com.jakev.testing.TestingActivity.test:()V
1a00 0700       |0000: const-string v0, "HERE" // string@0007
1a01 0100       |0002: const-string v1, "BSides Rulez" // string@0001
7120 0700 1000  |0004: invoke-static {v0, v1}, Landroid/util/Log;
                                .d:(Ljava/lang/String;Ljava/lang/String;)I // method@0007
0e00            |0007: return-void
```

In our app's DEX file

In another DEX file!

Resolving Trick

- **Question**: How do we call a method or use a string that a DexFile/DvmDex structure does not know about?
- **Answer**: Provide an index beyond the constant pool size, then add checks to dvm.*Resolver() function calls!
  - i.e. attempting to resolve string 8 out 7
  - Usually this indicates an error condition

Demo!

Future Plans & Research

- Move to Ice Cream Sandwich (4.0.0+)
- Expand hooking capabilities
  - All “payload” action handler
- More “man in the middle” capabilities
  - SQL database queries
  - Intent intercepting

Getting More Information

- Check my website & blog for updates, technical materials, etc.
  - http://www.thecobraden.com
  - http://blog.thecobraden.com
- Getting CobraDroid (beta)
  - http://www.thecobraden.com/projects/cobradroid
  - https://github.com/jakev/CobraDroidBeta (source)

The End

Feedback & Comments:
https://www.surveymonkey.com/s/BSidesDC13-Speaker