Google Android Platform
Introduction to the Android API, HAL and SDK

Zhaohui Wang, Angelos Stavrou
zwange@gmu.edu, astavrou@gmu.edu
George Mason University
What is Android?

“Android delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.”

-- http://android.com/about/
What is Android?

- A software stack, and nothing more
- Android was first released on Nov 12, 2007
- Latest Release, Android 2.2 on May 20, 2010, with Google’s Nexus One smart phone device
- MOST of the code under Apache License
- Linux based kernel, now on 2.6.32
- ARM based MSM (Qualcomm) chipset
- Graphics, Audio and other HAL implementations
What is Android?

- Development, debugging tools
- Dalvik JVM, (http://www.dalvikvm.com/)
- SDK available on 3 major OSes
- Incomplete/non standard GNU libraries/utils
- Includes “key mobile applications”, Google’s service highly integrated
The Genesis of Android?

- Open Handset Alliance:
  

- Google, eBay, OMRON, PacketVideo, ...
- ASUSTeK, HTC, LG, Garmin, Motorola, ...
- Sprint Nextel, T-Mobile, ...
- ARM, Atheros, Broadcom, Qualcomm, TI, ...

To date, more than 47 organizations
Noteworthy Features

- Android uses Java:
  - ... Everywhere, but only the mobile-appropriate bits!
  - “Android is almost but not quite Java(tm)”

- And so will you:
  - But nothing prevents native processes
  - Some native interfaces are available

- Broad Java support:
  - java.io;
  - java.security;
  - java.net;
  - java.sql;
Noteworthy Features

- **Strong security:**
  - Permissions-based
  - Applications sandboxed in separate VMs
  - Pervasive use of Linux process model

- **Built-in SQL:**
  - Property storage, retrieval
  - Utilized by nearly all standard components
  - Preferred, but not required

- **Specialized APIs:**
  - SurfaceFlinger
  - AudioFlinger
Noteworthy Features

- Highly-optimized Java implementation:
  - “Dalvik” VM implemented by Google
  - Custom bytecode format, processor model
  - Register-based, not stack-based

- Why?
  - “Didn’t want to pay Sun” (probably untrue)
  - Very memory- and performance-efficient
  - Highly tuned to limitations of small hardware

- Centralized object lifetime management:
  - Tied to component model
  - Tied to process model
  - Tied to user interface model
  - Tied to security model
Basic Terminology

- **Activity**:  
  - A single visual user interface component  
  - List of menu selections, icons, checkboxes, ...  
  - A reusable component

- **Service**:  
  - "Headless" activity component  
  - Background processes

- **Application**:  
  - Sequence of one or more Activities  
  - Manifest tells which Activity to run first  
  - Activities might come from other applications  
  - Not the Linux concept of "application"!
Basic Terminology

- **Task stack**:  
  - Sequences of application-centric Activity classes  
  - Foreground is visible to user  
  - BACK key returns to most-recent Activity

- **Broadcast receiver**:  
  - Component that receives announcements  
  - No user interface  
  - May launch an Activity in response

- **Content provider**:  
  - Provides application data to others  
  - The only way to share data
Power Management

- Obviously important!
  - Can be a difficult problem to solve
  - Too much model exposure is bad
  - Too little is also bad

- Extends the Linux device model:
  - Introduces “wake locks”
  - See android.os.PowerManager

- In a nutshell:
  - Applications don’t control power at all
  - Applications hold “locks” on power states
  - If no locks are held, Android powers down
Power Management

- PARTIAL_WAKE_LOCK
  - CPU on, screen off, keyboard off
  - Cannot power down via power button

- SCREEN_DIM_WAKE_LOCK
  - CPU on, screen dim, keyboard off

- SCREEN_BRIGHT_WAKE_LOCK
  - CPU on, screen bright, keyboard off

- FULL_WAKE_LOCK
  - CPU on, screen on, keyboard bright
**Example**

1. PowerManager pm =
2. (PowerManager) getSystemService(Context.POWER_SERVICE);
3. PowerManager.WakeLock wl =
4. pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, "tag");
5. wl.acquire();
6. // ..screen will stay on during this section..
7. wl.release();
Audio and Video APIs

- MediaPlayer class:
  - Standard support for many data formats
  - URI invokes appropriate input method
  - Consistent API regardless of data source

- MediaRecorder class:
  - Support for audio recording only
  - Video recording is “planned”

- Surfaceflinger:
  - Centralized framebuffer management
  - Related to 2D h/w acceleration

- Audioflinger:
  - Centralized audio stream management

You don’t work with these flingers directly!
Audio and Video APIs

Example

1. MediaPlayer mp = new MediaPlayer();
2. mp.setDataSource(PATH_TO_FILE);
3. mp.prepare();
4. mp.start();
5. mp.pause();
6. mp.stop();
Android Architecture
Android Package System

- APK files:
  - Package manifests
  - Classes
  - Dalvik bytecodes
  - Signatures, if any
The Hardware

- CPU: Qualcomm QSX8250 1Ghz
- Mother board: Qualcomm Mobile Station Modem (MSM) chipset, MSM7k series
- RAM: 512 MB
- ROM: 512 MB, partitioned as boot/system/userdata/cache
- External Storage: 4GB micro SD
- Audio Processor: msm_qdsp6 onboard processor, Firmware at /system/etc/vpimg
The Hardware

- Camera: Sensor_s5k3e2fx, 5 Megapixels
- Wifi+BlueTooth+FM: Boardcom BCM 4329, 802.11a/b/g/n, firmware at /system/etc/firmware/fw_bcm4329.bin
- Touch Screen Input: msm_ts touchscreen controller, capella
- Vibrator: Msm_vibrator on board vibrator
- Digital Compass: AK8973

The Hardware
System Initialization

- Bootloader: HBOOT-0.33.0012
- RADIO-4.02.02.14

1. kernel
2. Init.mahimahi.rc init.rc
3. debuggerd
4. AndroidRuntime
5. CameraService
6. System server(NetStat, Connectivity, WifiService, etc)
7. Zygote
8. Apps
General procedure:
- Get the code
  - 2.1GB (!) of git trees
  - Uses the repo tool to manage
- Build it
- Install it
- Tweaking and add your own code
- Build it and test it

- [http://android.git.kernel.org/](http://android.git.kernel.org/)
Building the Android Runtime

- `repo init -b froyo [eclair|donut|cupcake] -u git://android.git.kernel.org/platform/manifest.git`
- `repo sync`
  ...wait for 2.1GB code downloading
  ... apply tweaks ...
- `make [TARGET_PRODUCT=generic]`
Installing Android into a Target

- Build products:
  - userdata.img
  - ramdisk.img
  - system.img
  - kernel.img/boot.img

- And also:
  - out/target/product/<name>/root
  - out/target/product/<name>/system
  - out/target/product/<name>/data
Installing Android into a Target

- “What’s in there?”
  - The Android filesystem

- `# ls root`
  
  data/  init  init.rc  sys/
  default.prop  init.goldfish.rc  proc/  system/
  dev/  initlogo.rle  sbin/

- `# ls system`

  app/  build.prop  fonts/  lib/  usr/
  bin/  etc/  framework/  media/  xbin/
The Android SDK

- Key components:
  - Compilers, other tools
  - Documentation
  - Examples
  - Hardware emulator
  - Android Debug Bridge (adb)

Debugging your first Android App

- Configure USB connection, if you are working with devices

- Test adb and connect to device

  N:\android-sdk-windows\tools>adb devices
  * daemon not running. starting it now *
  * daemon started successfully *
  List of devices attached
  HT9CNP804091   device
  emulator-5556   device

- Launch a shell via adb:
  • The shell is actually on the target!

  N:\android-sdk-windows\tools>adb shell
  $
  $
Debugging your first Android App
Eclipse Android Plugin

- **Android Development Tool (ADT):**
  - Custom plugin for Eclipse IDE
- Helps automate:
  - Set up new Android projects
  - Create new applications, components
  - Debugging
- Install Eclipse, then:
  - Click *Help | Software Updates*...
  - [https://dl-ssl.google.com/android/eclipse/](https://dl-ssl.google.com/android/eclipse/)
  - Click *Install*...
- Then:
  - Point Eclipse to the Android SDK directory
  - *Window | Preferences | Android*

Your task

I. Get your helloworld running

II. Profiling and tracing your app like, what activities issued
Project Ideas

- Adore-ng rootkit porting on Android architecture
- Iphone OS Security Anatomy
- Iphone/Android MP3 decoder local exploit
- Your Idea (Brainstorming)
Recomanded Reading

- Understanding Android's Security Framework
  [http://siis.cse.psu.edu/android_sec_tutorial.html](http://siis.cse.psu.edu/android_sec_tutorial.html)
  A very good tutorial at CCS2008
- Mobile application Security on Android
### Boot.img layout (possible boot partition)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
<th>Address</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Magic: Android!</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>Kernel size in bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>Kernel_addr, phy ld addr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>Ramdisk size in bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>Ramdisk_addr, phy ld addr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.......</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel cmdline, 512 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timestamp/sha1 etc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x800</td>
<td>0 paddings</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kernel</td>
<td></td>
<td>Page aligned</td>
</tr>
<tr>
<td></td>
<td>ramdisk</td>
<td></td>
<td>Page aligned</td>
</tr>
<tr>
<td></td>
<td>Second optional</td>
<td></td>
<td>Page aligned</td>
</tr>
</tbody>
</table>

- First 2k page
- Page aligned
Nexus One in Memory Layout

- Base address 0x20000000, vary on other handset
- hdr.kernel_addr = base + 0x00008000;
- hdr.ramdisk_addr = base + 0x01000000;
- hdr.second_addr = base + 0x00F00000;
- hdr.tags_addr = base + 0x00000100;
Attack scenario

- Get the root privilege
- Remount filesystem as writable
  
  ```sh
mount -o rw,remount -t yaffs2 /dev/block/mtdblock0 /system
  ```
- DoS attacks
  1. Rmmod bcm4329; Rm /system/lib/modules/bcm4329.ko will disable the wireless network
Serial

- Host
  - mount -t usbfs usbfs /proc/bus/usb
  - lsusb -d 18d1:4e19 -v
  - Modprobe cdc_acm
  - modprobe usbserial vendor=0x18d1 product=0x4e11
  - Pay attention to the ttyUSBxx, the last one is what you need, and you may launch adb before this, otherwise, usbserial will override the driver.
  - stty ispeed 9600 ospeed 9600 -F /dev/ttyUSB1

- Device
  - Enable ttyfs/ttyGS0 device on your phone, by adding kgdboc=ttyGS0,9600 in the boot.img kernel booting command line
  - Modify the kernel default enable table,
<table>
<thead>
<tr>
<th>Dev</th>
<th>size</th>
<th>erase</th>
<th>size</th>
<th>name</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtd0</td>
<td>000e0000</td>
<td>00020000</td>
<td>&quot;misc&quot;</td>
<td>0x000003ee0000-0x000003fc0000</td>
<td></td>
</tr>
<tr>
<td>mtd1</td>
<td>00500000</td>
<td>00020000</td>
<td>&quot;recovery&quot;</td>
<td>0x000004240000-0x000004740000</td>
<td></td>
</tr>
<tr>
<td>mtd2</td>
<td>00280000</td>
<td>00020000</td>
<td>&quot;boot&quot;</td>
<td>0x000004740000-0x0000049c0000 2.5M</td>
<td></td>
</tr>
<tr>
<td>mtd3</td>
<td>09100000</td>
<td>00020000</td>
<td>&quot;system&quot;</td>
<td>0x0000049c0000-0x00000dac0000</td>
<td></td>
</tr>
<tr>
<td>mtd4</td>
<td>05f00000</td>
<td>00020000</td>
<td>&quot;cache&quot;</td>
<td>0x00000dac0000-0x0000139c0000</td>
<td></td>
</tr>
<tr>
<td>mtd5</td>
<td>0c440000</td>
<td>00020000</td>
<td>&quot;userdata&quot;</td>
<td>0x0000139c0000-0x00001fe0000</td>
<td></td>
</tr>
</tbody>
</table>

- Where is radio???
  - 0-0x000003ee0000, 62.875MB
  - 0x000003fc0000 -0x000004240000, 2.5MB
kgdb

- CONFIG_KGDB
- CONFIG_KGDB_SERIAL_CONSOLE
- CONFIG_MAGIC_SYSRQ=y
  CONFIG_DEBUG_INFO=y
  CONFIG_FRAME_POINTER =y (recommended, but really useful?)
  # CONFIG_DEBUG_RODATA is not set
- echo kgdbts=V2 > /sys/module/kgdbts/parameters/kgdbts
- KGDB_TESTS_ON_BOOT
- echo ttyfs0,9600 > /sys/module/kgdboc/parameters/kgdboc
- Press and hold the Alt key, and the press and release the SysRq. While still holding alt, press the g key, then release all the keys.
Serial over USB

Device
- Minicomm or any other application talk to /dev/ttyXXX
- Device-side Linux OS
- Gadget Serial Driver (fserial.ko)
- USB Periph. Controller Driver
- USB Stack

Host
- Minicomm or HyperTerm
- Host-side Linux/Windows OS
- CDC ACM or generic USB serial driver
- USB Host Controller Driver (hcd)
- USB Host Controller Driver
- USB Stack

USB cable
GDB setup

- cd ~/mydroid
- . Build/envsetup.sh
- Lunch 1
- emulator -verbose -show-kernel -netfast

emulator: control console listening on port 5556, ADB on port 5557
GDB setup

- telnet localhost 5556
  - In telnet, type: redir add tcp:10000:10000
  - Press `CTRL-]` and, at the `telnet>` prompt, type: `quit`
- `adb shell gdbserver 10.0.2.15:10000 --attach <PID of program>`
- `adb shell gdbserver 10.0.2.15:10000 Binary`

- arm-eabi-gdb
  - `out/target/product/generic/symbols/system/bin/app_process`

Reading symbols from
  `/root/mydroid/out/target/product/generic/symbols/system/bin/app_process...done.`
GDB setup

- In *gdb*:
  - set solib-search-path
    out/target/product/generic/symbols/system/lib:
    out/target/product/generic/symbols/system/bin
  - target remote localhost:10000

- Debugging is an art….
- GDB Cheat sheet:
GDB setup (kernel)

- `emulator -verbose -show-kernel -netfast -kernel /root/mydroid/kernels/android/arch/arm/boot/zImage`
- `qemu –monitor telnet::6666,server &`

QEMU waiting for connection on: `telnet::6666,serve`

- `telnet localhost 6666`
  - QEMU 0.10.50 monitor - type 'help' for more information
  - (qemu)
GDB setup (kernel)

- arm-eabi-gdb ~/mydroid/kernels/NexusOne/vmlinux
- target remote localhost:1234

[New Thread 1]
0xafe09ec4 in ?? ()
Disassemble zImage file

- Forensic before you actually run it
- arm-eabi-objdump -EL -b binary -D -m armv5t zImage | grep 8b1f
- 3456: 35fc: 088b1f00 stmeq fp, {r8, r9, sl, fp, ip}
- hexdump -C zImage | grep '1f 8b 08'
- hexdump -C zImage | grep 'the kernel'
- 864: 000035f0 74 68 65 20 6b 65 72 6e 65 6c 2e 0a 00 1f 8b 08 |the kernel......|
- Align to 1f 8b 08 00,
- dd if=zImage of=piggy.gz bs=1 skip=13821(35fc+1)
- gunzip piggy.gz
- strings piggy | grep version
Flash splash

- [http://www.gotontheinter.net/logo.rle](http://www.gotontheinter.net/logo.rle)
- `D:\android-sdk-windows\tools>fastboot flash splash1 splash.raw565`
- sending 'splash1' (750 KB)... OKAY
- writing 'splash1'... INFOsignature checking...
- FAILED (remote: signature verify fail)
Bootloader unlocking

- Fastboot oem unlock
- Dslsrv.gmu.edu/isa673/fastboot.zip