



# Android Internals

## Lecture 3

Security of Mobile Devices

2018



**SMD**

Android Architecture

Linux Kernel

Binder

Android Framework

Managers

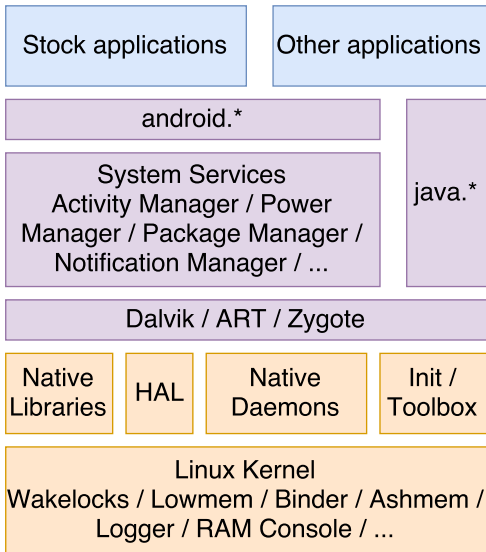
Android Architecture

Linux Kernel

Binder

Android Framework

Managers



Android Architecture

Linux Kernel

Binder

Android Framework

Managers

- ▶ "Androidized" kernel
- ▶ Hundreds of patches over the standard kernel
- ▶ Device-specific functionality, fixes, enhancements
- ▶ Android Mainlining Project / Android Upstreaming Project
- ▶ Many features get into the mainline kernel

- ▶ Wakelocks
- ▶ Low-Memory Killer
- ▶ Binder
- ▶ Anonymous Shared Memory
- ▶ Alarm
- ▶ RAM Console
- ▶ Paranoid Networking

- ▶ On desktops and laptops
  - ▶ The user decides when the system goes to sleep
- ▶ The Android kernel goes to sleep as often as possible
- ▶ Sometimes you want to keep the system from going to sleep
  - ▶ Input from the user, critical operations
- ▶ Wakelocks keep the system awake



- ▶ A wakelock must be obtained by the application when it needs to stay awake
  - ▶ Apps use abstractions that handle locking
  - ▶ Apps can request wakelocks directly from PowerManager Service
  - ▶ Device drivers call in-kernel wakelock primitives
- ▶ Equivalent included in mainline, from Linux 3.5
  - ▶ Autosleep
  - ▶ `epoll()` flag `EPOLLWAKEUP`

- ▶ Linux has Out-of-memory (OOM) killer
- ▶ Low-Memory Killer:
  - ▶ Prevents the activation of the OOM killer
  - ▶ Kills processes with components unused for a long time
  - ▶ System unlikely to run out of memory
  - ▶ Included in mainline, from Linux 3.10

- ▶ Based on OOM adjustments mechanism
  - ▶ Different OOM kill priorities for different processes
- ▶ The userspace may control OOM killing policies
- ▶ Policies applied at startup by init
- ▶ Modified and enforced by Activity Manager

- ▶ Levels assigned to processes based on their components
  - ▶ Levels from -17 to 15 (high -> killed)
- ▶ Threshold (MinFree) for each type of process
  - ▶ Foreground\_app - application in foreground
  - ▶ Visible\_app - visible but not in foreground
  - ▶ Secondary\_server - service
  - ▶ Hidden\_app - hidden, needed by a running app
  - ▶ Content\_provider - provide data
  - ▶ Empty\_app - app not active
- ▶ Starts killing when the threshold is reached

- ▶ IPC mechanism
- ▶ SysV IPC can lead to resource leakage in the kernel (vulnerability)
- ▶ File-based, reference-counted
- ▶ Similar to POSIX SHM, differences:
  - ▶ Uses reference counting to destroy the memory regions
  - ▶ Shrink mapped regions when the system needs memory
  - ▶ To shrink a region it must be unpinned

- ▶ First process creates region, uses Binder to share descriptor with other processes
- ▶ System services rely on ashmem, through IMemory interface
  - ▶ Surface Flinger, Audio Flinger
- ▶ Driver included in the staging tree from Linux 3.3

- ▶ Uses the RTC and HRT functionalities
- ▶ `setitimer()`
  - ▶ Generate a signal when the time expires
  - ▶ Based on HRT
  - ▶ Does not work when the system is suspended
  - ▶ The application receives the signal when the device wakes up

- ▶ Using RTC, the alarm will be fired even if the system is suspended
  - ▶ RTC hardware device
  - ▶ `/dev/rtc` - `ioctl()` calls
- ▶ Uses HRT by default
- ▶ When the system is about to suspend, it uses RTC
- ▶ Apps use alarms even when the system is suspended



- ▶ `/dev/alarm` character device, `ioctl()`
- ▶ `SystemClock`, `AlarmManager` class rely on the driver
  - ▶ `SystemClock` - obtain and set time
  - ▶ `AlarmManager` - provide alarms to apps
- ▶ The driver and `AlarmManager` use `WakeLocks`
  - ▶ The app that receives the alarm runs before the system is suspended again
- ▶ Included in mainline, from Linux 3.20

- ▶ Standard Linux
  - ▶ Processes are allowed to create sockets and access the network
- ▶ Android
  - ▶ Restrict access to the network
  - ▶ Based on the group of the caller process

- ▶ AID\_INET - AF\_INET and AF\_INET6 sockets
  - ▶ `android.permission.INTERNET`
- ▶ AID\_NET\_RAW - raw INET sockets
- ▶ AID\_NET\_ADMIN - configuration of network interfaces and routing tables
  - ▶ `android.permission.NET_ADMIN`

- ▶ AID\_NET\_BT - Bluetooth sockets
  - ▶ `android.permission.BLUETOOTH`
- ▶ AID\_NET\_BT\_ADMIN - configure Bluetooth
  - ▶ `android.permission.BLUETOOTH_ADMIN`

- ▶ Association between permissions and GIDs
- ▶ /etc/permissions/platform.xml

```
<permission name="android.permission.BLUETOOTH_ADMIN" >
    <group gid="net_bt_admin" />
</permission>
<permission name="android.permission.BLUETOOTH" >
    <group gid="net_bt" />
</permission>
<permission name="android.permission.INTERNET" >
    <group gid="inet" />
</permission>
<permission name="android.permission.NET_ADMIN" >
    <group gid="net_admin" />
</permission>
```

Android Architecture

Linux Kernel

**Binder**

Android Framework

Managers

- ▶ RPC mechanism
- ▶ Initially in BeOS (then bought by Palm)
- ▶ OpenBinder project
- ▶ OpenBinder developers working in Android team
- ▶ Android Binder does not derive from OpenBinder
  - ▶ Clean re-write of the same functionality
- ▶ OpenBinder documentation for understanding the mechanism
- ▶ Binder driver in the mainline from kernel 3.19

- ▶ Remote object invocation
  - ▶ Remote services as objects
  - ▶ Interface definition and reference to it
- ▶ Cornerstone of Android architecture
  - ▶ Apps talk to systems services
  - ▶ Apps talk to application services
- ▶ Developers don't use the Binder directly
- ▶ Use interfaces and stubs generated with the aidl tool
- ▶ Public API uses stubs to communicate with system services



- ▶ Part of the Binder implemented in a kernel driver
- ▶ Character device
- ▶ `/dev/binder`
- ▶ `ioctl()` calls
- ▶ Transmit parcels of data (serialized) between entities

Android Architecture

Linux Kernel

Binder

**Android Framework**

Managers

- ▶ On top of the native userspace
- ▶ `android.*` packages, System Services, Android Runtime
- ▶ Code in `frameworks/` directory in AOSP
- ▶ Key building blocks: Service Manager, Dalvik/ART, Zygote

- ▶ Form an object-oriented OS on top of Linux
- ▶ System Server
  - ▶ All components run in the `system_server` process
  - ▶ Many Java-based services/managers, 2 C-based services
  - ▶ Power Manager, Activity Manager, Location Manager, etc.
  - ▶ Surface Flinger, Sensor Service (C/C++)
- ▶ Media Server
  - ▶ `mediaserver` process
  - ▶ C/C++ code
  - ▶ Audio Flinger, Media Player Service, Camera Service

- ▶ Default before Android 5.0
- ▶ Java VM optimized for mobile architectures
  - ▶ Lower memory footprint
- ▶ Works with `.dex` files instead of `.jar` files
  - ▶ 50% smaller
- ▶ Incompatible with Java bytecode
- ▶ Register based, not stack based
- ▶ 16 bit instructions instead of 8 bit instructions (stack)
- ▶ Less instructions and higher execution speed

- ▶ Includes Just-in-Time (JIT) compiler
  - ▶ From Android 2.2
  - ▶ ARM, x86, MIPS
  - ▶ Profiles the applications at runtime
  - ▶ Translates segments of bytecode (traces) into machine instructions
  - ▶ Code runs directly on the CPU, not one instruction at a time by the VM
  - ▶ The rest of the bytecode interpreted by Dalvik
  - ▶ Performance improvements

- ▶ Available from Android 4.4
- ▶ Default from Android 5.0
- ▶ Dalvik Executable format
- ▶ Ahead-of-Time compilation (AoT)
  - ▶ dex2oat tool
  - ▶ Translate the dex file into an executable for the target device
  - ▶ At installation time
  - ▶ Replaces JIT compilation and Dalvik interpretation
  - ▶ Installation takes longer
  - ▶ Executables occupy storage space
  - ▶ Additional verifications

- ▶ Improved garbage collection
  - ▶ More efficient
- ▶ Support for sampling profiler
  - ▶ Does not affect app performance
- ▶ More debugging features
  - ▶ Especially for monitoring and GC
- ▶ More details in case of exceptions and crash reports



- ▶ Daemon used to launch apps
- ▶ Parent of all processes
- ▶ Preloads in RAM all Java classes and resources needed by apps
- ▶ Listens to connections on its socket for requests to start apps
  - ▶ `/dev/socket/zygote`
- ▶ When it gets a request, it forks itself and launches the app

- ▶ Copy-on-write (COW)
- ▶ Classes and resources are not modified, so all apps use them from Zygote
  - ▶ A single version of classes and resources in RAM
- ▶ The System Server is started explicitly by Zygote
- ▶ The PPID of all apps is the PID of Zygote

- ▶ From Android 5.0
- ▶ Logd daemon
- ▶ Centralized user-mode logger
- ▶ Addresses the disadvantages of circular buffers
- ▶ Integration with SELinux
  - ▶ Registers as auditd
  - ▶ Receive messages via netlink

- ▶ Uses 4 sockets
- ▶ `/dev/socket/logd` - control
- ▶ `/dev/socket/logdw` - write-only
- ▶ `/dev/socket/logdr` - read-only
- ▶ Unnamed netlink socket - SELinux

- ▶ Write log messages:
  1. Log class
  2. Liblog library
  3. /dev/socket/logdw socket
  
- ▶ Read log messages:
  1. logcat
  2. Liblog library
  3. /dev/socket/logdr socket

Android Architecture

Linux Kernel

Binder

Android Framework

**Managers**

- ▶ Performs system service handle lookups
- ▶ The Yellow pages book of all system services
- ▶ A service must be registered to the Service Manager to be available
- ▶ Started by `init` before any other service
- ▶ Opens `/dev/binder` and becomes the Context Manager of the Binder
- ▶ Binder ID 0 = "magic object" = Service Manager

- ▶ System Server registers every service with the Service Manager
- ▶ Any component that wants to talk to a system service:
  - ▶ Asks the Service Manager for a handle
  - ▶ `getSystemService()`
  - ▶ Invokes the methods of the service using the handle
- ▶ Only to access system services
- ▶ Used by the `dumpsys` utility to obtain the status of the system services



- ▶ One of the most important services in the System Server
- ▶ Handles activity lifecycle
- ▶ Sends intents
- ▶ Starts new components (activities, services)
- ▶ Obtains content providers
- ▶ Responsible with the Application Not Responding (ANR) messages
- ▶ Involved in
  - ▶ Permission checks
  - ▶ OOM adjustments for the Low-Memory Killer
  - ▶ Task management

- ▶ Starts the Launcher (with `Intent.CATEGORY_HOME`)
- ▶ When an app is started from Launcher
  - ▶ Launcher's `onClick()` callback is called
  - ▶ Launcher calls the `startActivity()` from `ActivityManager` (through Binder)
  - ▶ `ActivityManager` calls `startViaZygote()` method
  - ▶ Opens socket to Zygote and asks to start the activity
- ▶ `am` command for invoking the functionality of the `ActivityManager`
- ▶ `isUserAMonkey()`

- ▶ Manages the `.apk` files in the systems
- ▶ API for installing, uninstalling, upgrading `.apk` files
- ▶ Works with files located in `/data/system/`
  - ▶ `packages.xml`
  - ▶ `packages.list`
- ▶ `system_server` and `installd` processes
- ▶ Resolves intents
- ▶ `pm` command for invoking the functionality of the `PackageManager`

- ▶ Control the power state of the device
- ▶ Handles WakeLocks
- ▶ Includes the WakeLock class
  - ▶ `acquire()`, `release()`
- ▶ Apps request WakeLocks from PowerManager
- ▶ All calls to the Power Management (kernel) go through PowerManager
- ▶ Can force device to go to sleep
- ▶ Set the brightness of the backlights

- ▶ Karim Yaghmour, "Embedded Android: Porting, Extending, and Customizing", Chapter 2
- ▶ Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, Georg Wicherski, "Android Hacker's Handbook", Chapter 2
- ▶ <https://wiki.linaro.org/LMG/Kernel/Upstreaming>
- ▶ <https://source.android.com/devices/tech/dalvik/>

- ▶ Linux kernel
- ▶ WakeLocks
- ▶ Low-Memory killer
- ▶ Binder
- ▶ Ashmem
- ▶ Alarm
- ▶ Logger
- ▶ System Server
- ▶ Dalvik
- ▶ ART
- ▶ Zygote
- ▶ Service Manager
- ▶ Activity Manager
- ▶ Package Manager
- ▶ Power Manager