Aurasium: Practical Policy Enforcement for Android Applications

Rubin Xu
University of Cambridge

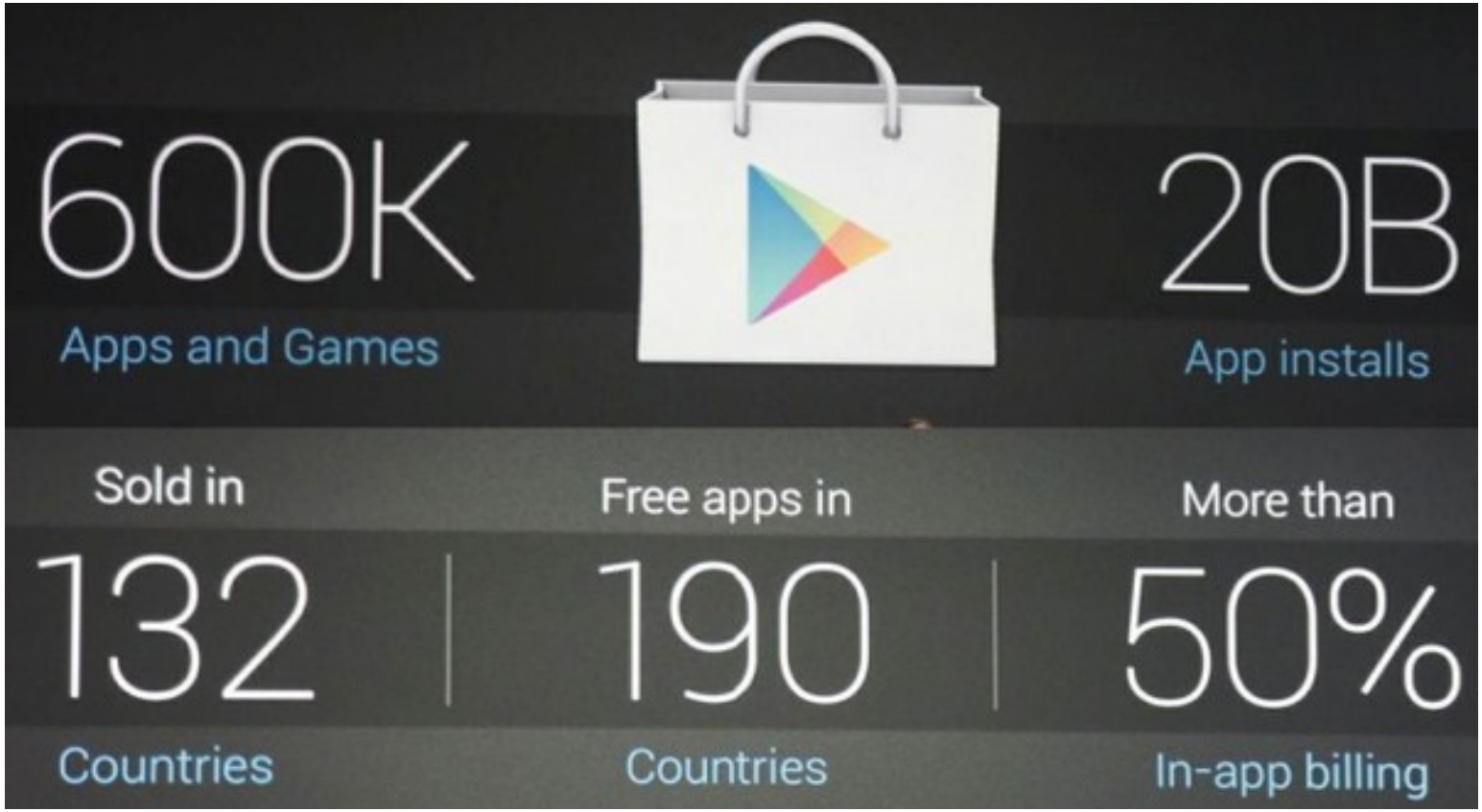
Hassen Saidi
SRI International

Ross Anderson
University of Cambridge

USENIX Security Symposium 2012

Goal

- Address the multiple threats posed by malicious applications on Android



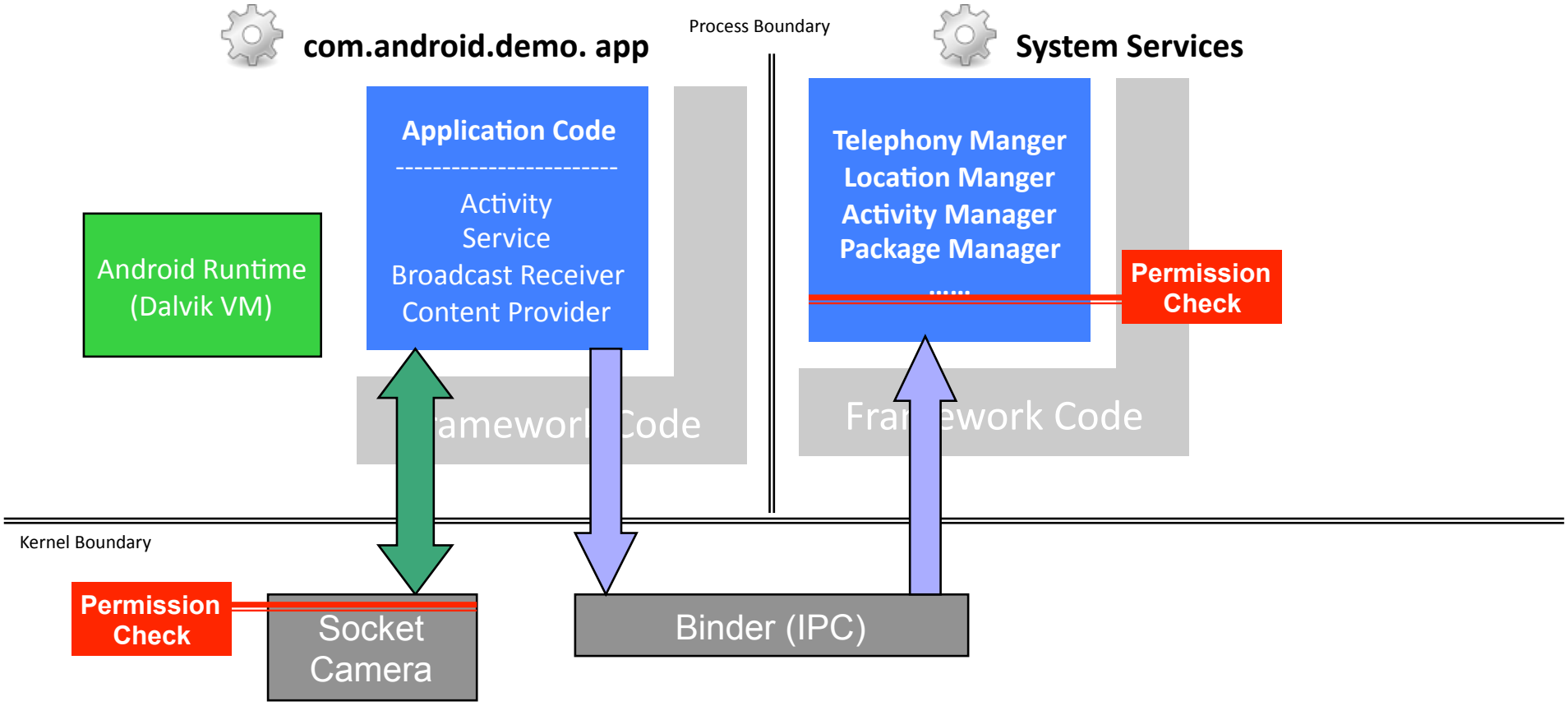
Android Malicious Apps

Introduction to Android

- Security Features
  - Process Isolation
  - Linux user/group permission
  - App requests permission to OS functionalities
    - Most checked in remote end i.e. system services
    - A few (Internet, Camera) checked in Kernel, as special user group

Introduction to Android

- Security Features



Malicious Android Apps

- Abuse permissions:
  - Permissions are granted for as long as an App is installed on a device
  - No restrictions on how often resources and data are accessed
- Access and transmit private data
- Access to malicious remote servers
- application-level privilege escalation
  - Confused deputy attacks
- Gain root privilege

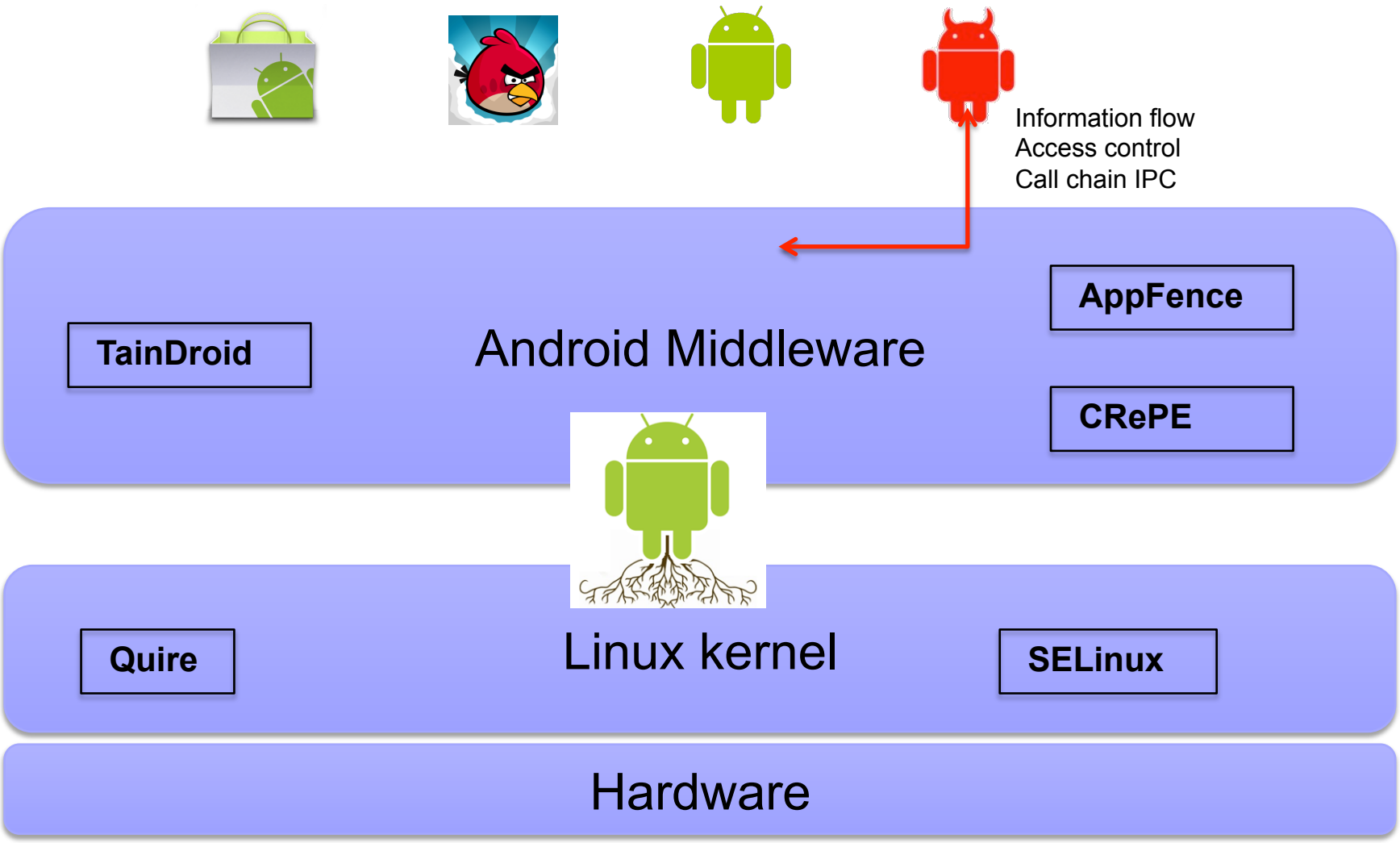
Alternative Approaches

- App vetting: Google's Bouncer
  - 40% decrease in malware
  - Ineffective once App installed on the device
- AV products:
  - Scanning
  - Have no visibility into the runtime of an App
- Fine grain permissions checking
  - Require modifications to the OS
- Virtualization
  - Require modification to the OS

Related work

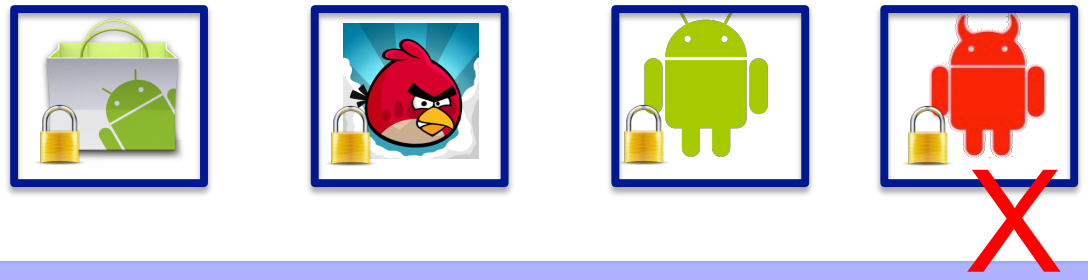
- Existing Work
  - TaintDroid (OSDI 10)
  - CRePE (ISC 10)
  - AppFence (CCS 11)
  - Quire (USENIX Security 2011)
  - SELinux on Android
  - Taming Privilege-Escalation (NDSS 2012)
- Limitations
  - Modify OS – requires rooting and flashing firmware.

Related Approaches



Solution: Aurasium

Repackage Apps to intercept all Interactions with the OS



Information flow
Access control
Call chain IPC
and many more!

Android Middleware

Linux kernel

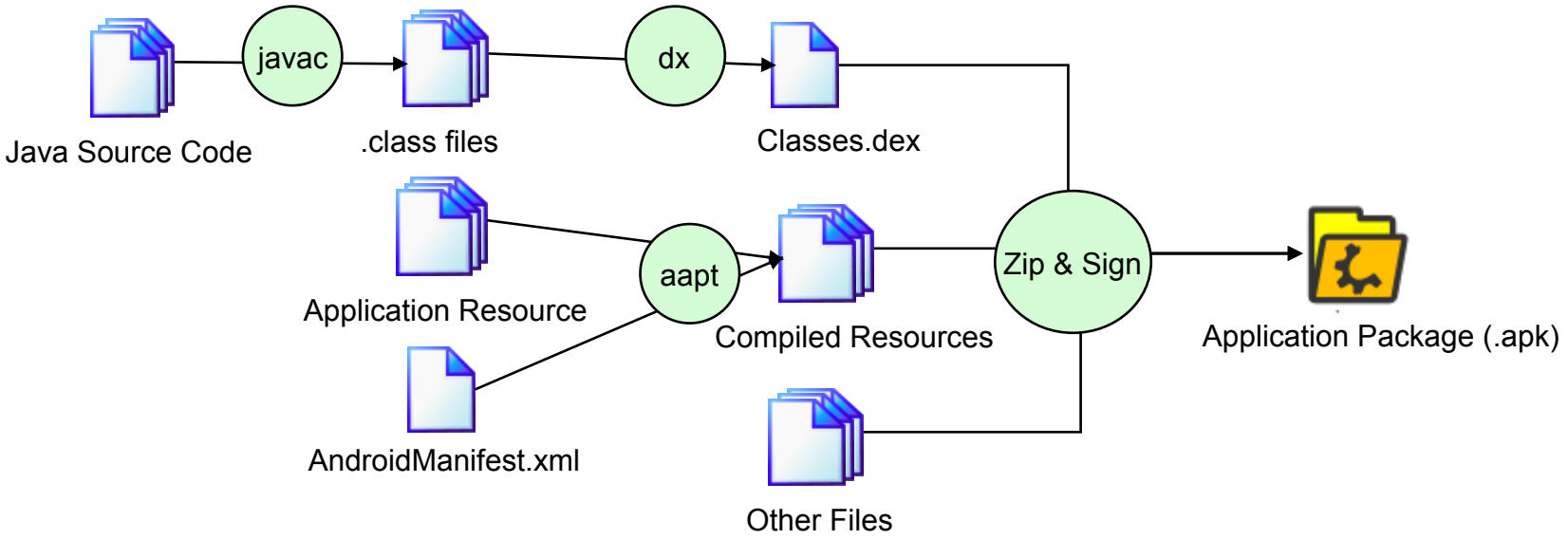
Hardware

Aurasium Internals

- Two Problems to Solve
  - Introducing alien code to arbitrary application package
  - Reliably intercepting application interaction with the OS

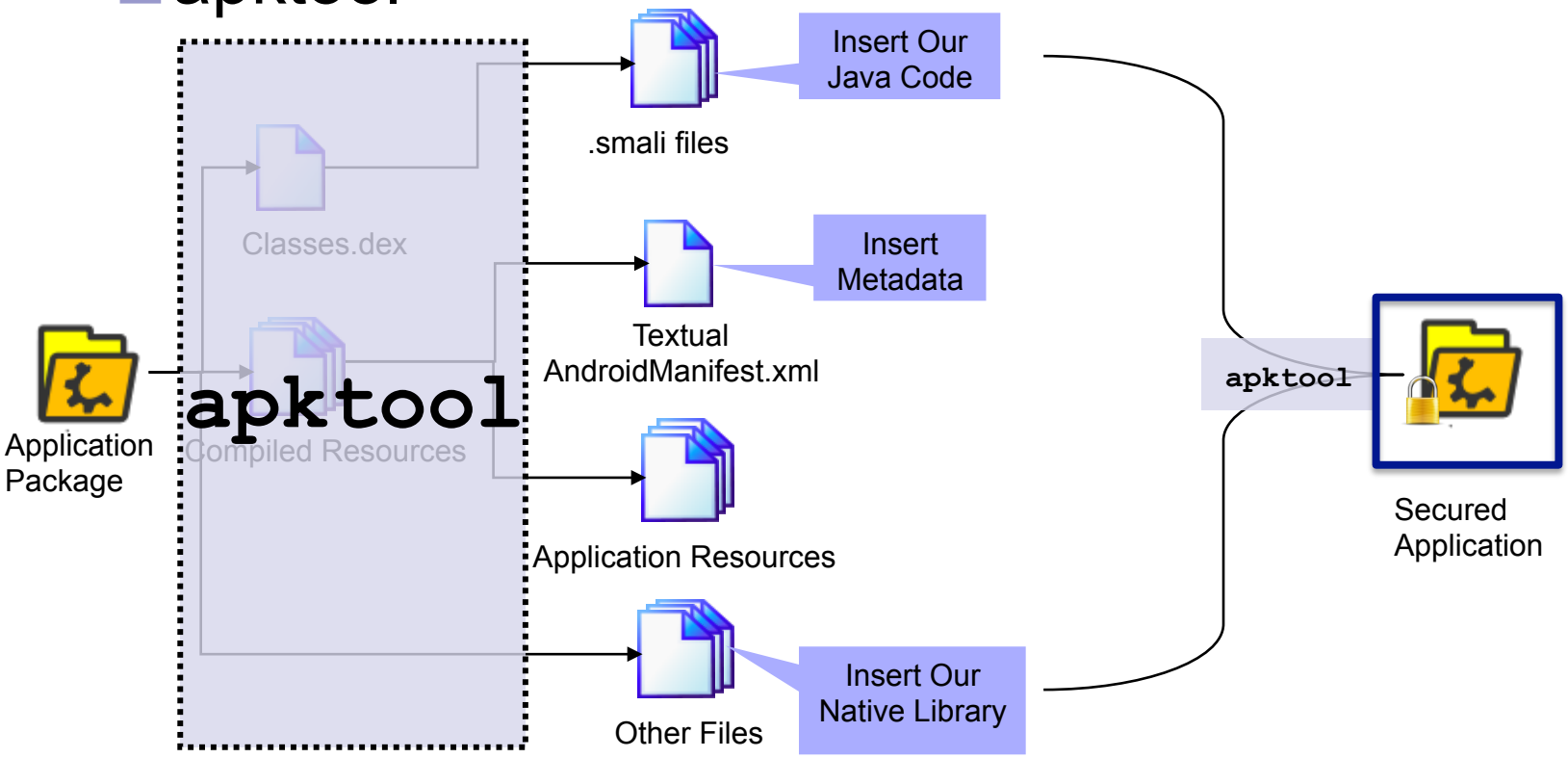
Aurasium Internals

- How to add code to existing applications
  - Android application building and packaging process



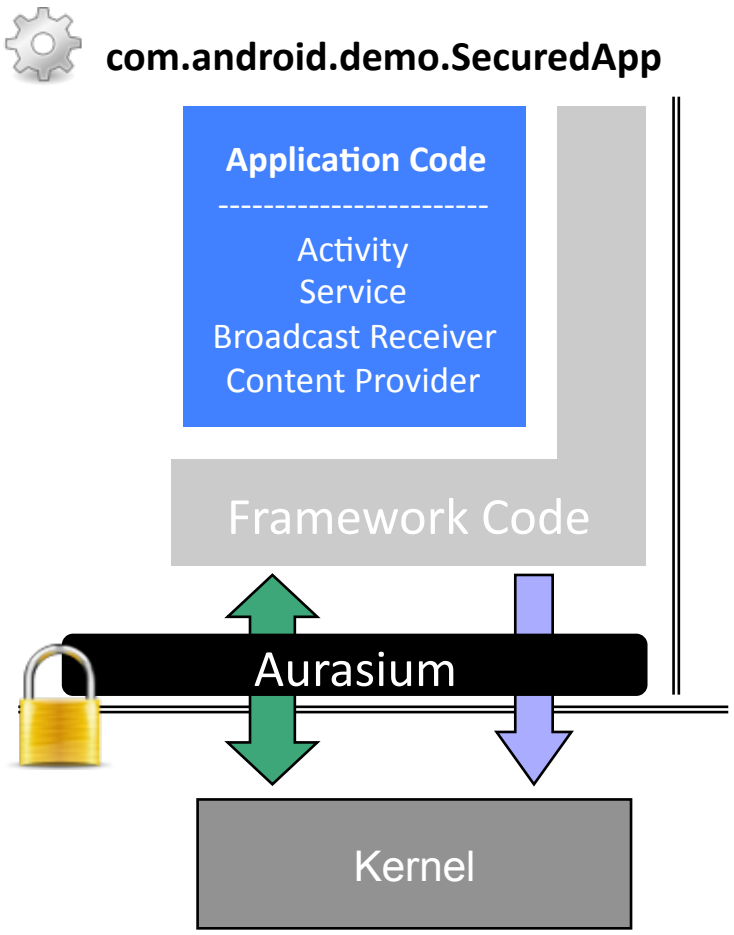
Aurasium Internals

- How to add code to existing applications
  - apktool



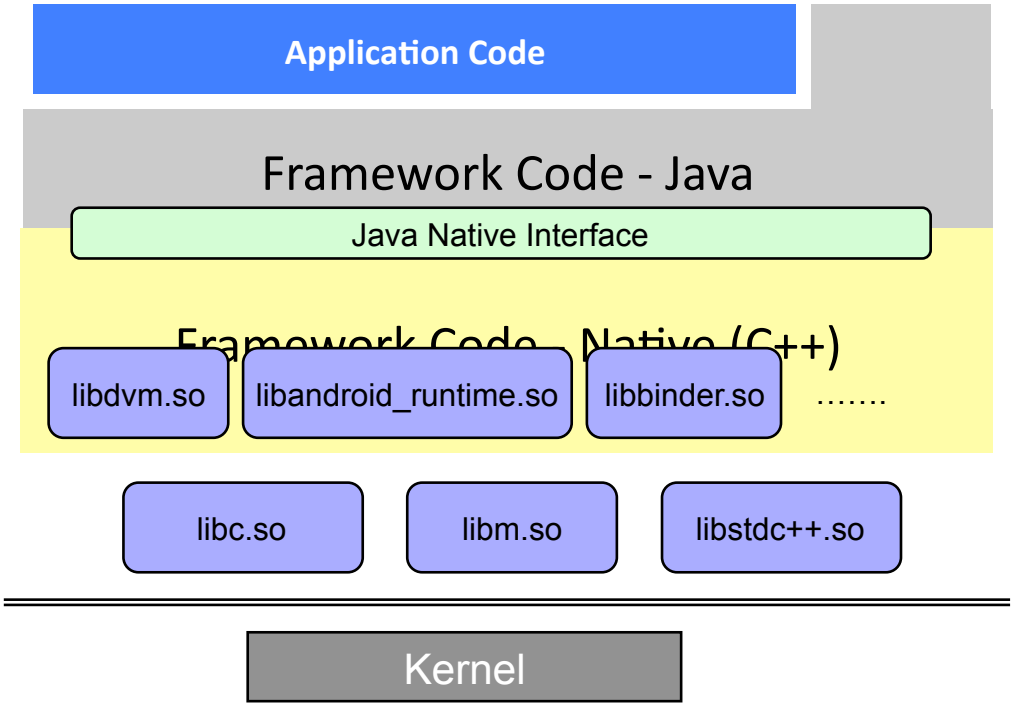
Enforcing Security & Privacy Policy

- Aurasium way
  - Per-application basis
  - No need to root phone and flash firmware
  - Almost non-bypassable



Aurasium Internals

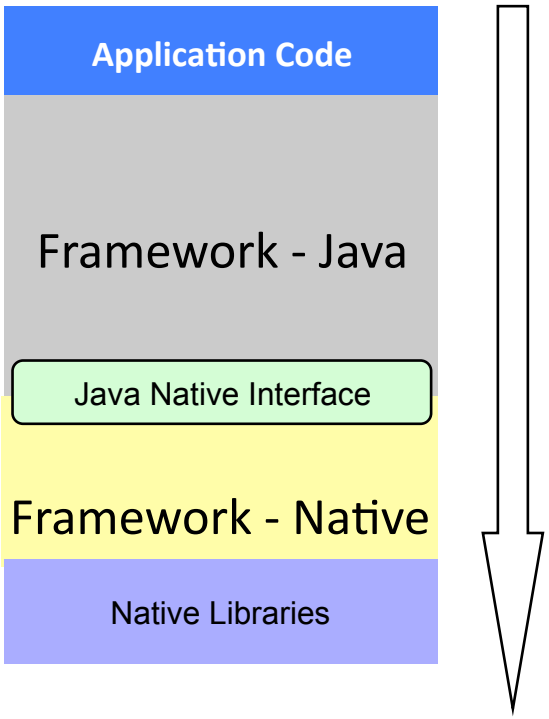
- How to Intercept
  - A closer look at app process




```
00008000-00009000 r-xp 00000000 1f:01 570        /system/bin/app_process
00009000-0000a000 rwxp 00001000 1f:01 570        /system/bin/app_process
0000a000-00264000 rwxp 00000000 00:00 0          [heap]
......
41d6c000-41f7b000 r-xs 00000000 1f:01 935        /system/framework/core.jar
41f7b000-42443000 r-xp 00000000 1f:04 290        /data/dalvik-cache/system@framework@core.jar@classes.dex
42443000-42488000 rwxp 00000000 00:00 0
42488000-424c2000 r-xs 00000000 1f:01 932        /system/framework/ext.jar
424c2000-4254e000 r-xp 00000000 1f:04 291        /data/dalvik-cache/system@framework@ext.jar@classes.dex
4254e000-427f9000 r-xs 00000000 1f:01 936        /system/framework/framework.jar
427f9000-42e56000 r-xp 00000000 1f:04 292        /data/dalvik-cache/system@framework@framework.jar@classes.dex
4321b000-43224000 rwxp 00000000 00:00 0
43224000-4323c000 r-xs 000a3000 1f:01 1107       /system/app/Vending.apk
4323c000-43290000 rwxp 00000000 00:00 0
......
ac700000-ac714000 r-xp 00000000 1f:01 744        /system/lib/libsurfaceflinger_client.so
ac900000-ac919000 r-xp 00000000 1f:01 692        /system/lib/libpixelflinger.so
aca00000-aca91000 r-xp 00000000 1f:01 608        /system/lib/libdvm.so
aca98000-aca9a000 rwxp 00000000 00:00 0
ad100000-ad12e000 r-xp 00000000 1f:01 639        /system/lib/libnativehelper.so
ad300000-ad379000 r-xp 00000000 1f:01 732        /system/lib/libandroid_runtime.so
ad381000-ad387000 rwxp 00000000 00:00 0
.....
afb00000-afb16000 r-xp 00000000 1f:01 810        /system/lib/libm.so
afc00000-afc01000 r-xp 00000000 1f:01 716        /system/lib/libstdc++.so
afd00000-afd41000 r-xp 00000000 1f:01 734        /system/lib/libc.so
afd44000-afd4f000 rwxp 00000000 00:00 0
b0001000-b000c000 r-xp 00001000 1f:01 583        /system/bin/linker
b000d000-b0016000 rwxp 00000000 00:00 0
beb71000-beb7d000 rwxp 00000000 00:00 0          [stack]
00008000-00009000 r-xp 00000000 1f:01 570        /system/bin/app_process
00009000-0000a000 rwxp 00001000 1f:01 570        /system/bin/app_process
0000a000-00264000 rwxp 00000000 00:00 0          [heap]
......
41d6c000-41f7b000 r-xs 00000000 1f:01 935        /system/framework/core.jar
41f7b000-42443000 r-xp 00000000 1f:04 290        /data/dalvik-cache/system@framework@core.jar@classes.dex
42443000-42488000 rwxp 00000000 00:00 0
42488000-424c2000 r-xs 00000000 1f:01 932        /system/framework/ext.jar
424c2000-4254e000 r-xp 00000000 1f:04 291        /data/dalvik-cache/system@framework@ext.jar@classes.dex
4254e000-427f9000 r-xs 00000000 1f:01 936        /system/framework/framework.jar
427f9000-42e56000 r-xp 00000000 1f:04 292        /data/dalvik-cache/system@framework@framework.jar@classes.dex
4321b000-43224000 rwxp 00000000 00:00 0
43224000-4323c000 r-xs 000a3000 1f:01 1107       /system/app/Vending.apk
4323c000-43290000 rwxp 00000000 00:00 0
......
ac700000-ac714000 r-xp 00000000 1f:01 744        /system/lib/libsurfaceflinger_client.so
ac900000-ac919000 r-xp 00000000 1f:01 692        /system/lib/libpixelflinger.so
aca00000-aca91000 r-xp 00000000 1f:01 608        /system/lib/libdvm.so
aca98000-aca9a000 rwxp 00000000 00:00 0
ad100000-ad12e000 r-xp 00000000 1f:01 639        /system/lib/libnativehelper.so
ad300000-ad379000 r-xp 00000000 1f:01 732        /system/lib/libandroid_runtime.so
ad381000-ad387000 rwxp 00000000 00:00 0
.....
afb00000-afb16000 r-xp 00000000 1f:01 810        /system/lib/libm.so
afc00000-afc01000 r-xp 00000000 1f:01 716        /system/lib/libstdc++.so
afd00000-afd41000 r-xp 00000000 1f:01 734        /system/lib/libc.so
afd44000-afd4f000 rwxp 00000000 00:00 0
b0001000-b000c000 r-xp 00001000 1f:01 583        /system/bin/linker
b000d000-b0016000 rwxp 00000000 00:00 0
beb71000-beb7d000 rwxp 00000000 00:00 0          [stack]
```

Aurasium Internals

- How to Intercept
  - Example: Socket Connection




```
ApkMonitorActivity.onClick()
```

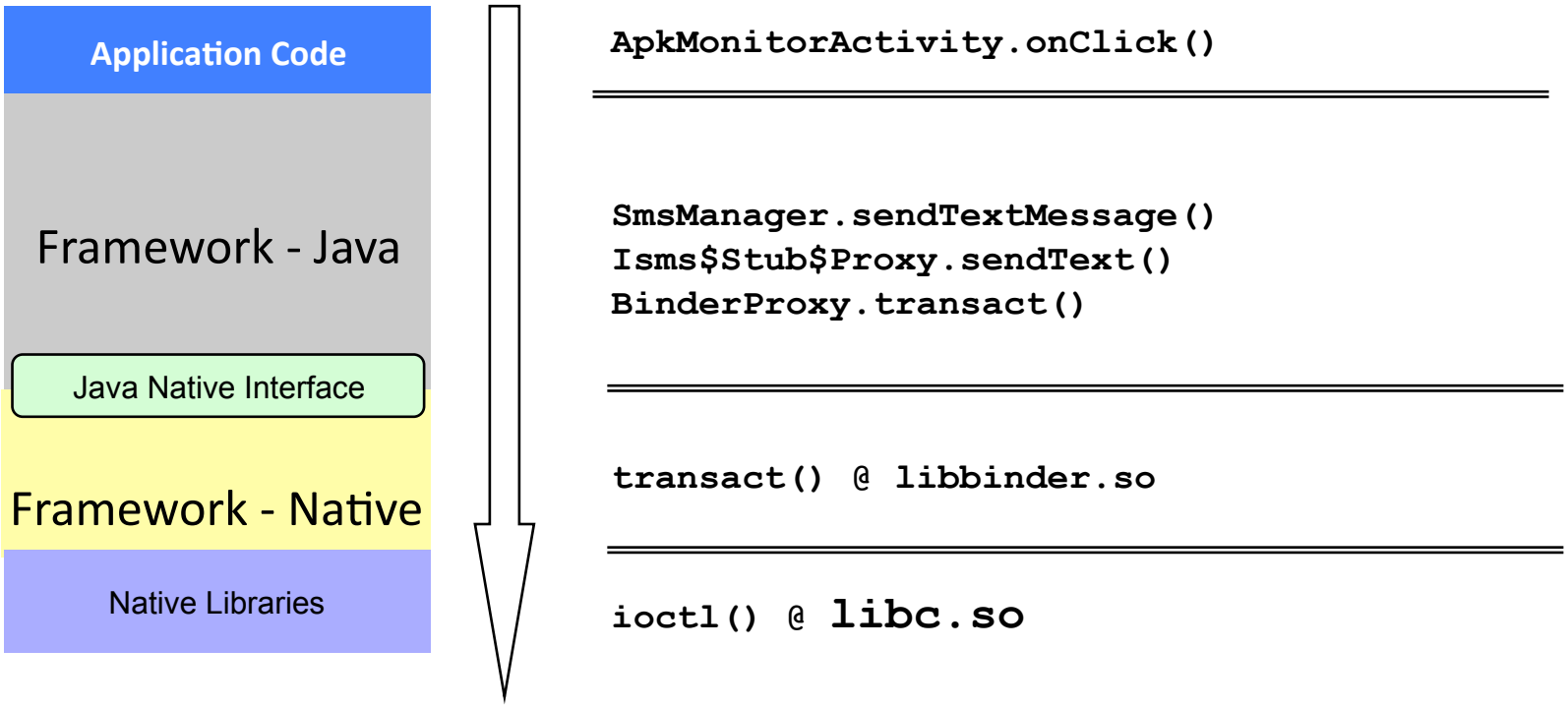
```
HttpURLConnectionImpl.makeConnection()
HttpConnection.<init>()
Socket.connect()
PlainSocketImpl.connect()
OSNetworkSystem.connect()
```

```
OSNetworkSystem_connect() @ libnativehelper.so
```

```
connect() @ libc.so
```

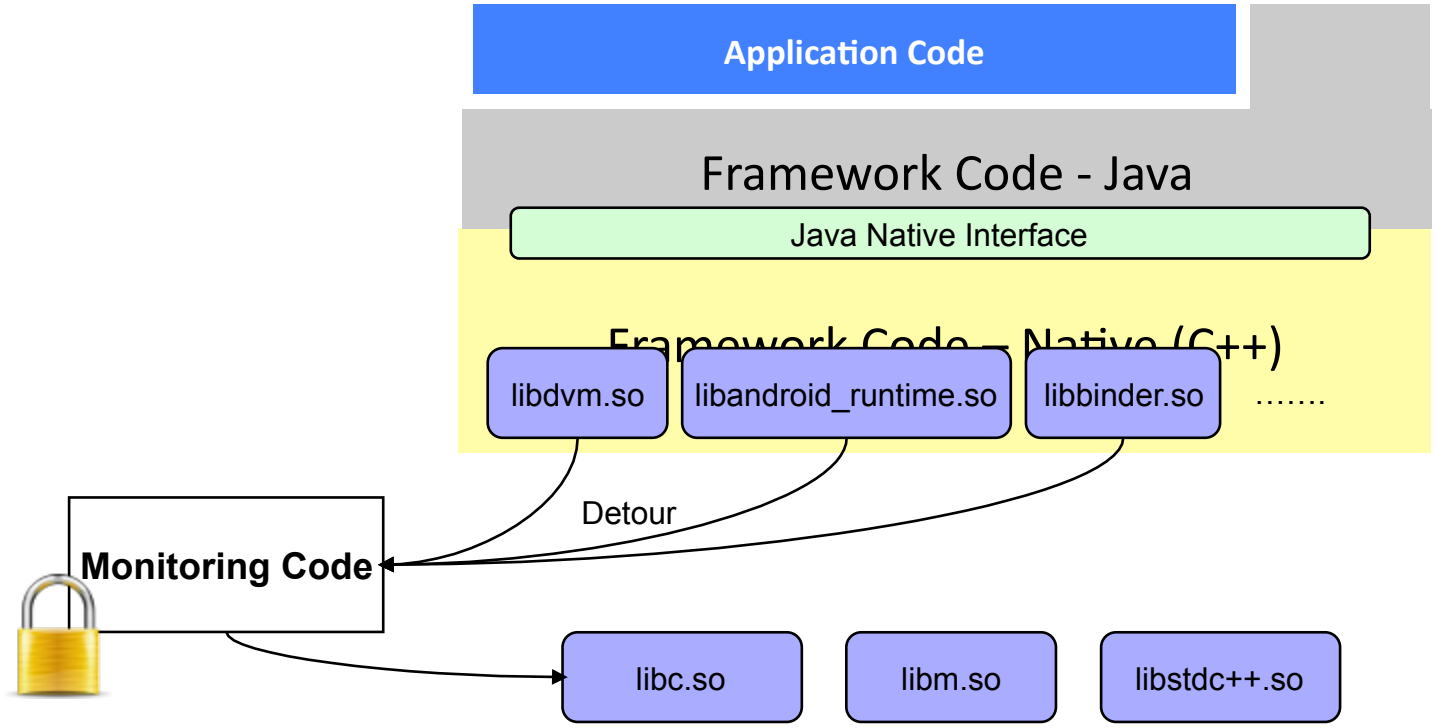
Aurasium Internals

- How to Intercept
  - Example: Send SMS



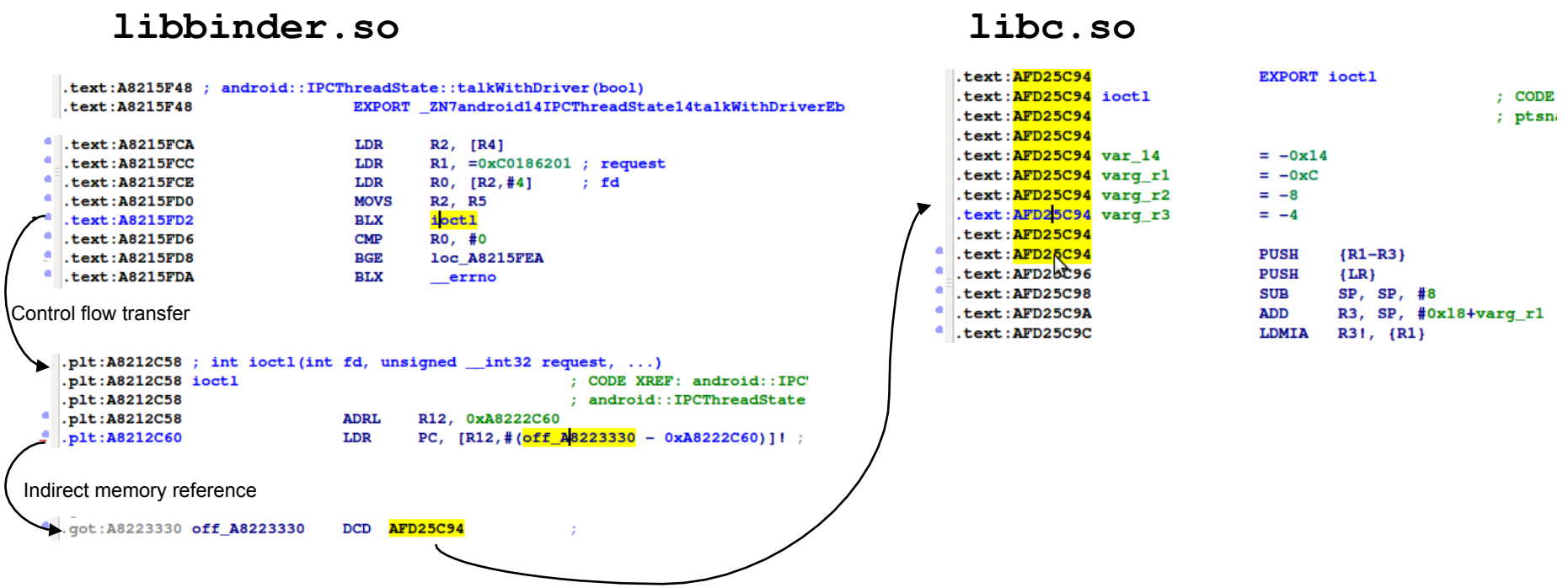
Aurasium Internals

- How to Intercept
  - Intercept at lowest boundary – libc.so



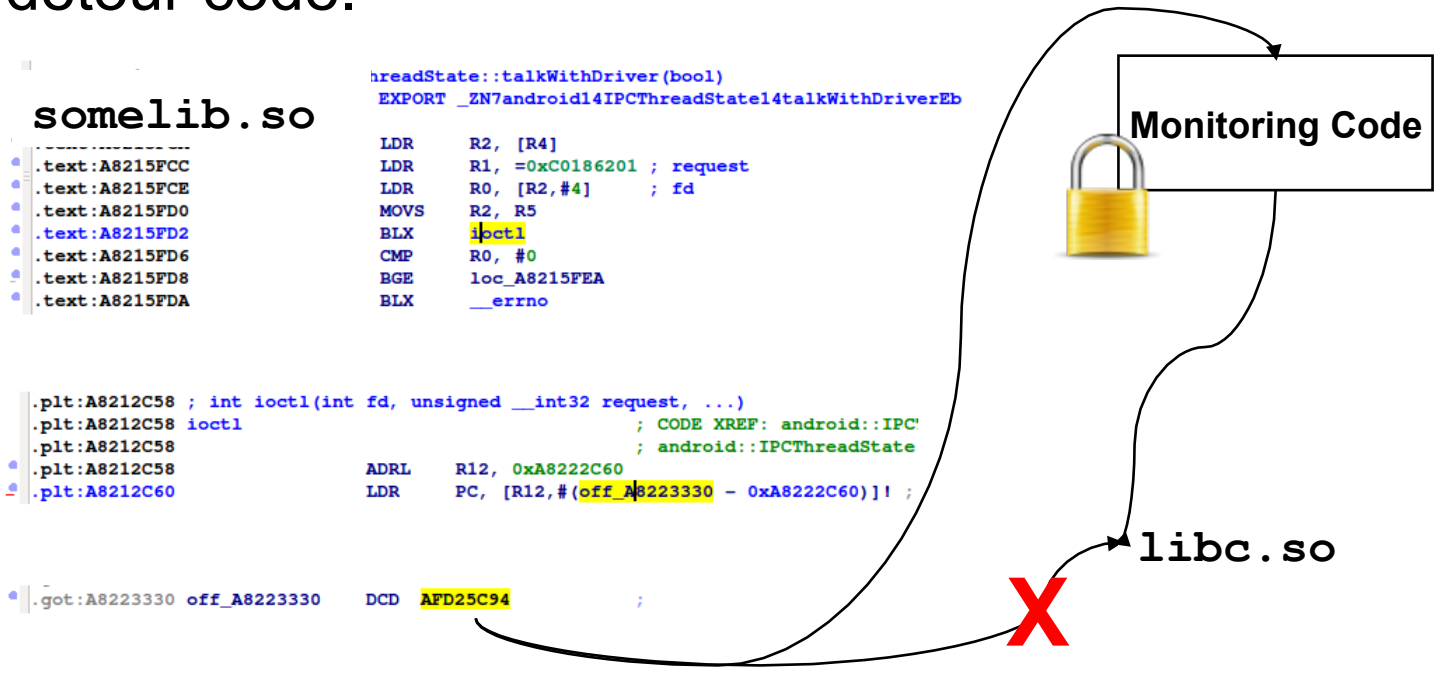
Aurasium Internals

- How to Intercept
  - Look closer at library calls - dynamic linking



Aurasium Internals

- How to Intercept
  - Key: Dynamically linked shared object file
  - Essence: Redo dynamic linking with pointers to our detour code.



Aurasium Internals

- How to Intercept
  - Implemented in native code
  - Almost non-bypassable
    - Java code cannot modify arbitrary memory
    - Java code cannot issue syscall directly
    - Attempts to load native code is monitored
      - dlopen()

What can you do with Aurasium?

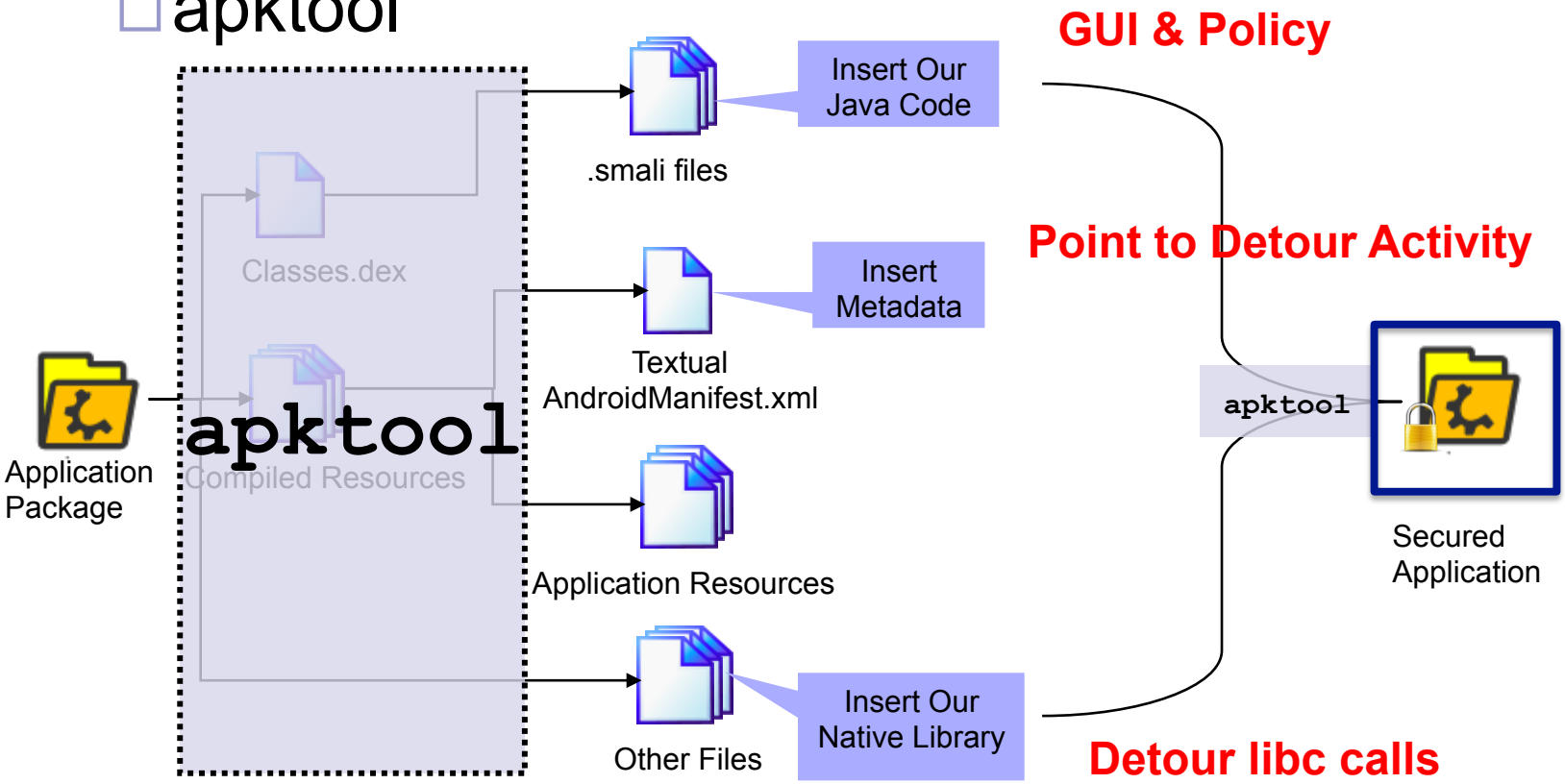
- Total visibility into the interactions of an App with the OS and other Apps
  - Internet connections
    - connect()
  - IPC Binder communications
    - ioctl()
  - File system manipulations
    - write(), read()
  - Access to resources
    - loctl(), read, write()
  - Linux system calls
    - fork(), execvp()

Aurasium Internals

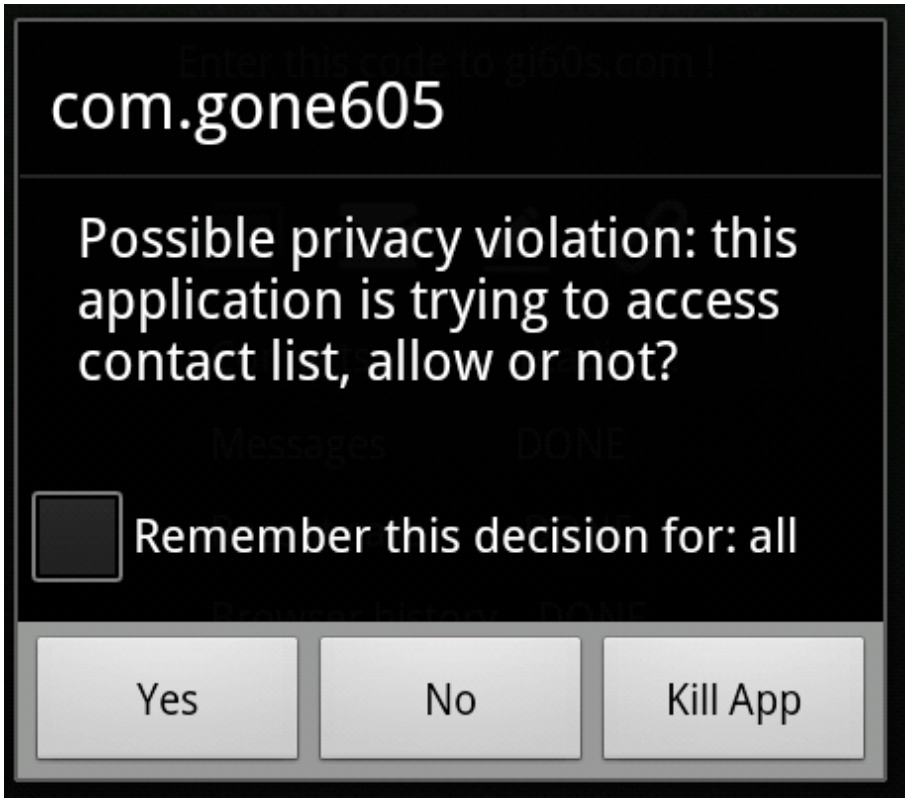
- How to add code to existing applications
  - Inevitably destroy original signature
    - In Android, signature = authorship
  - Individual app not a problem

Aurasium Internals

- How to add code to existing applications
  - apktool



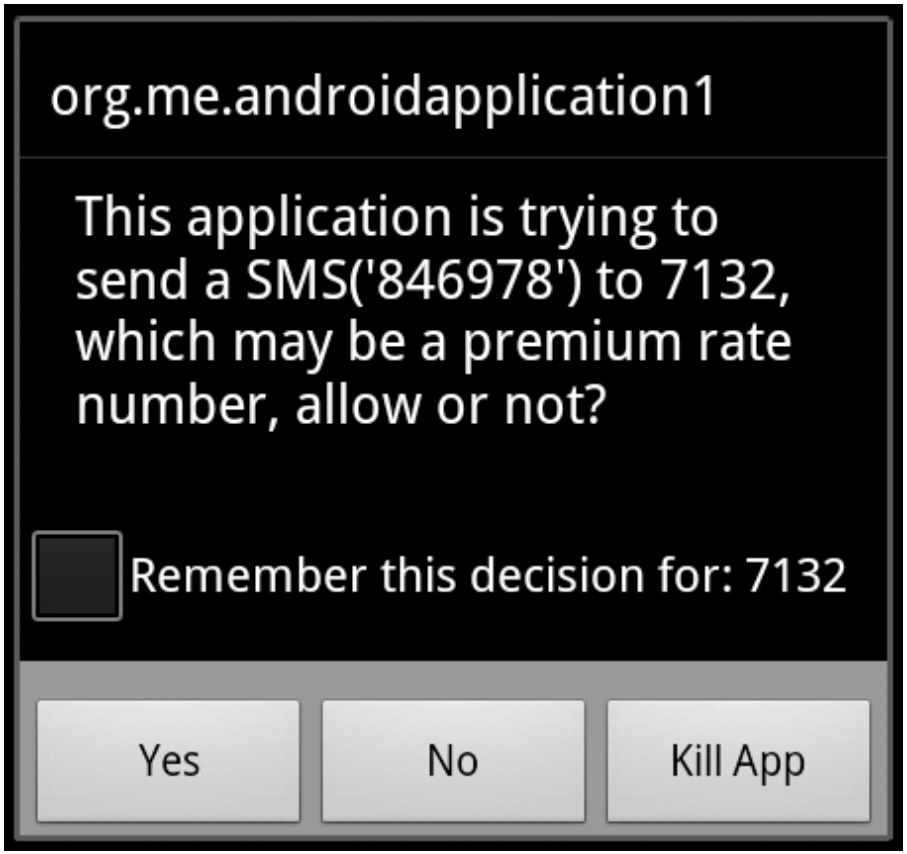
Evaluation


com.gone605
Possible privacy violation: this application is trying to access contact list, allow or not?
Remember this decision for: all
Yes
No
Kill App

Evaluation


com.moregame.drakula
Possible privacy violation: this application is trying to access your location information, allow or not?
Remember this decision for: last position
Yes
No
Kill App

Evaluation


org.me.androidapplication1
This application is trying to send a SMS('846978') to 7132, which may be a premium rate number, allow or not?
Remember this decision for: 7132
Yes
No
Kill App

Evaluation


com.km.launcher
This application is connecting to remote server: su.5k3g.com:80, allow or not?
Remember this decision for: su.5k3g.com:80
IP Address: 218.66.104.242
Threat Level: Unverified
Country: CHINA
Yes
No
Kill App

Evaluation


com.km.launcher
This application may be trying to increase its control over your phone by executing su, allow or not?
Remember this decision for: su
Yes
No
Kill App

Evaluation

- Tested on Real-world Apps
  - 3491 apps from third-party application store.
  - 1260 malware corpus from Android Genome.
  - Results
    - Repackaging:
      - 3476/1258 succeed (99.6%/99.8%)
      - Failure mode: apktool/baksmali assembly crashes
    - Device runs
      - Nexus S under Monkey – UI Exerciser in SDK
      - Intercept calls from all of 3189 runnable application.

Limitations

- 99.9% is not 100%
  - Rely on robustness of apktool
  - Manual edit of Apps as a workaround
- Native code can potentially bypass Aurasium:
  - Already seen examples of native code in the wild that is capable of doing so
  - Some mitigation techniques exist

Conclusion

- New approach to Android security/privacy
- Per-app basis, no need to root phone
- Tested against many real world apps
- Have certain limitations

The End

- Try it out at www.aurasium.com