The Firewall Android Deserves: A Context-aware Kernel Message Filter and Modifier

...

David Wu

Agenda

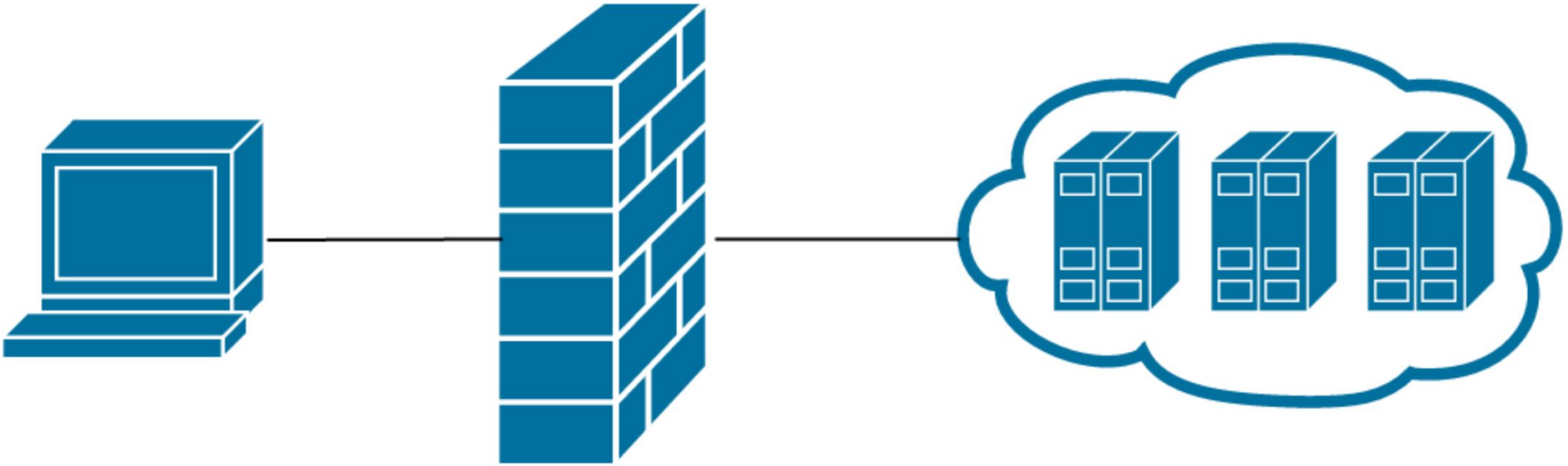
- Overview of project
- Android security background
- Binder IPC
- BinderFilter
- Logging and analysis tools
- Picky
- Demos
- Discussion & future work
- Questions
- Slides: https://goo.gl/2SlB40

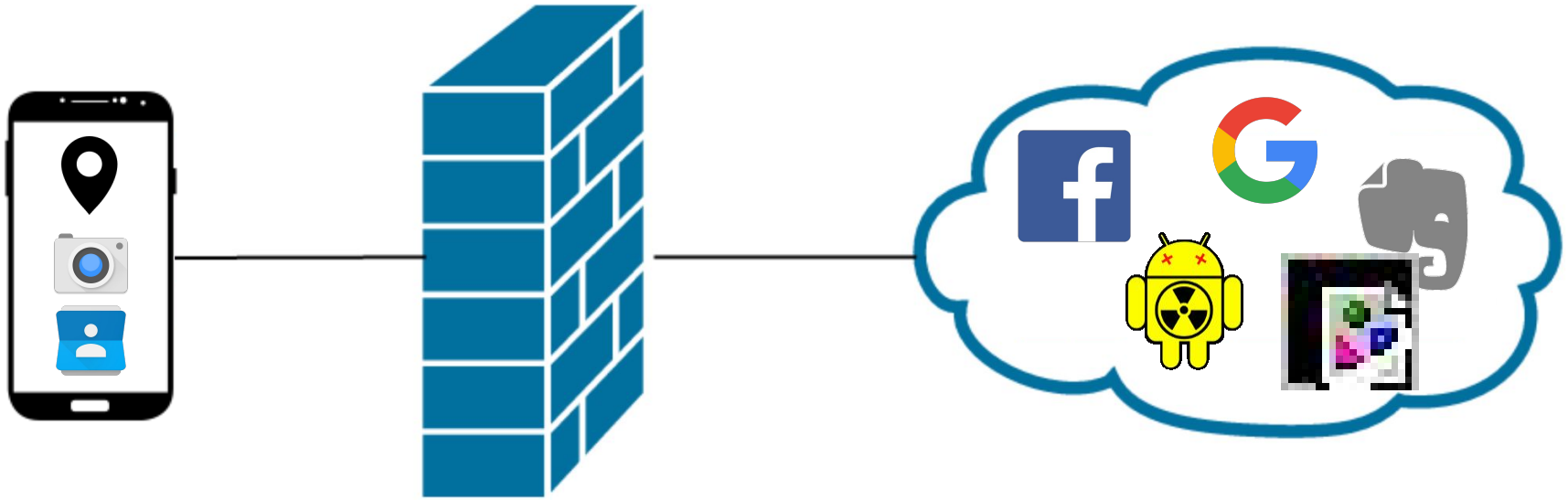
Who am I?

- Graduated June 2016, Dartmouth College
- OpenSSH and Android security research with Sergey Bratus
- Web analysis automation and Android security research at Ionic Security
- Particle physics simulations at Brookhaven National Lab

Motivation

- Dynamic (run-time) blocking of all inter-app communication
- Context informed policy decisions
- Binder message parser and hook





Previous Research

- rovo89. **Xposed.** 2016
- Stephan Heuser, Adwait Nadkarni, William Enck, Ahmad-Reza Sadegi. **Boxify.** 2015
- Nitay Artenstein and Idan Revivo. **Man in the Binder.** 2014
- Xueqiang Wang, Kun Sun, Yuewu Wang, Jiwu Jing. **DeepDroid.** 2015
- Mauro Conti, Vu Thein Nguyen, Bruno Crispo. **CRePE.** 2011
- Android Marshmallow. **Google**. 2015

Project Overview

- Inter-application message firewall and Binder hooking framework
  - Linux kernel driver, C
- Binder IPC message parser and formatter
  - Script, Python
- User policy generation
  - Android application, Java & C (JNI, NDK)
- https://github.com/dxwu/AndroidBinder
- https://github.com/dxwu/Picky

Features

- Complete mediation
  - Everything is done in the kernel Binder IPC system
- Dynamic permission blocking for all applications
- Blocking of custom, user-specified messages at runtime
- Contextual blocking
  - Wifi state, Wifi SSID, Bluetooth state, Apps running
- Modification of message data
  - Camera, Location
- Usable interface for setting policy

Permissions

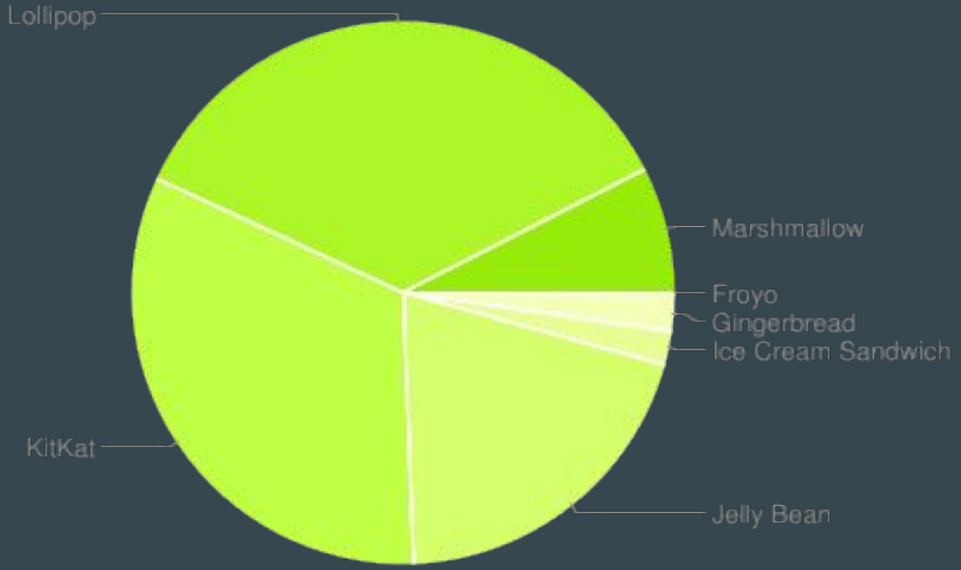
android.permission.CAMERA
android.permission.RECORD_AUDIO
android.permission.READ_CONTACTS
android.permission.WRITE_CONTACTS
android.permission.GET_ACCOUNTS
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_COARSE_LOCATION
android.permission.READ_EXTERNAL_STORAGE
android.permission.WRITE_EXTERNAL_STORAGE
com.android.vending.
INTENT_PACKAGE_INSTALL_COMMIT
android.permission.INTERNET
android.permission.SYSTEM_ALERT_WINDOW
android.permission.WRITE_SETTINGS
android.permission.READ_PHONE_STATE
android.permission.CALL_PHONE
android.permission.READ_CALL_LOG
android.permission.WRITE_CALL_LOG
android.permission.SEND_SMS
android.permission.RECEIVE_SMS
android.permission.READ_SMS
android.permission.RECEIVE_MMS
android.permission.RECEIVE_WAP_PUSH
android.permission.READ_CALENDAR
android.permission.WRITE_CALENDAR
android.permission.BODY_SENSORS
android.permission.ACCESS_NETWORK_STATE
android.permission.CHANGE_NETWORK_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.CHANGE_WIFI_STATE
android.permission.BATTERY_STATS
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.NFC
android.permission.FLASHLIGHT
com.android.browser.permission.READ_HISTORY_BOOKMARKS
android.permission.TRANSMIT_IR
android.permission.USE_SIP

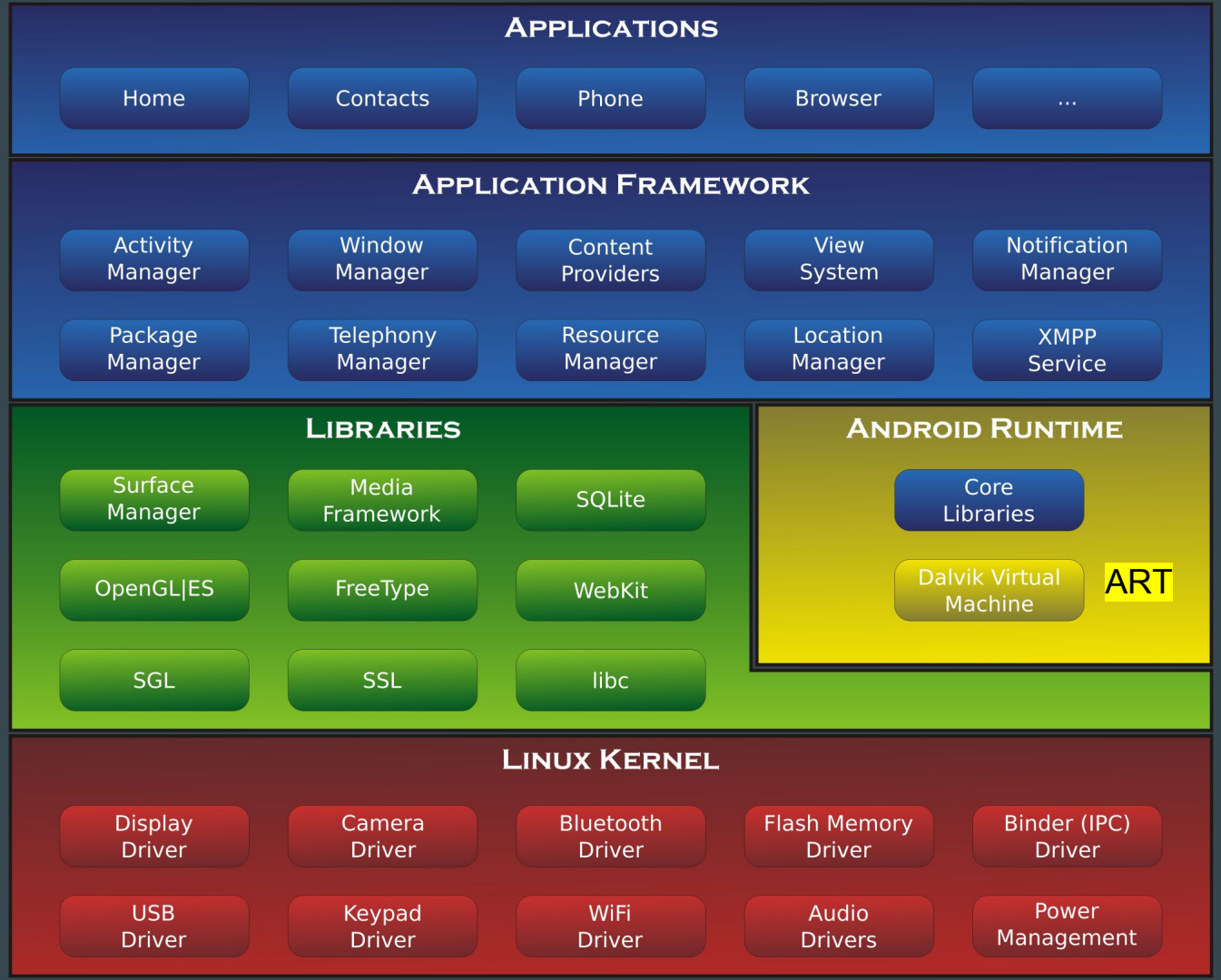
Installation methods

- Android versions 4.3+ have disabled loadable kernel modules
  - Kernel make config does not set CONFIG_MODULES=y
- To place a hook in Binder, which is a statically compiled kernel driver, we have to recompile the kernel sources with our modifications
- Flash new kernel image onto Android with fastboot
  - This preserves user information, apps, and state!
- Requirements:
  - Linux build env (Include headers don't work on OSX)
  - adb, fastboot, abootimg
  - Unlocked bootloader, root access

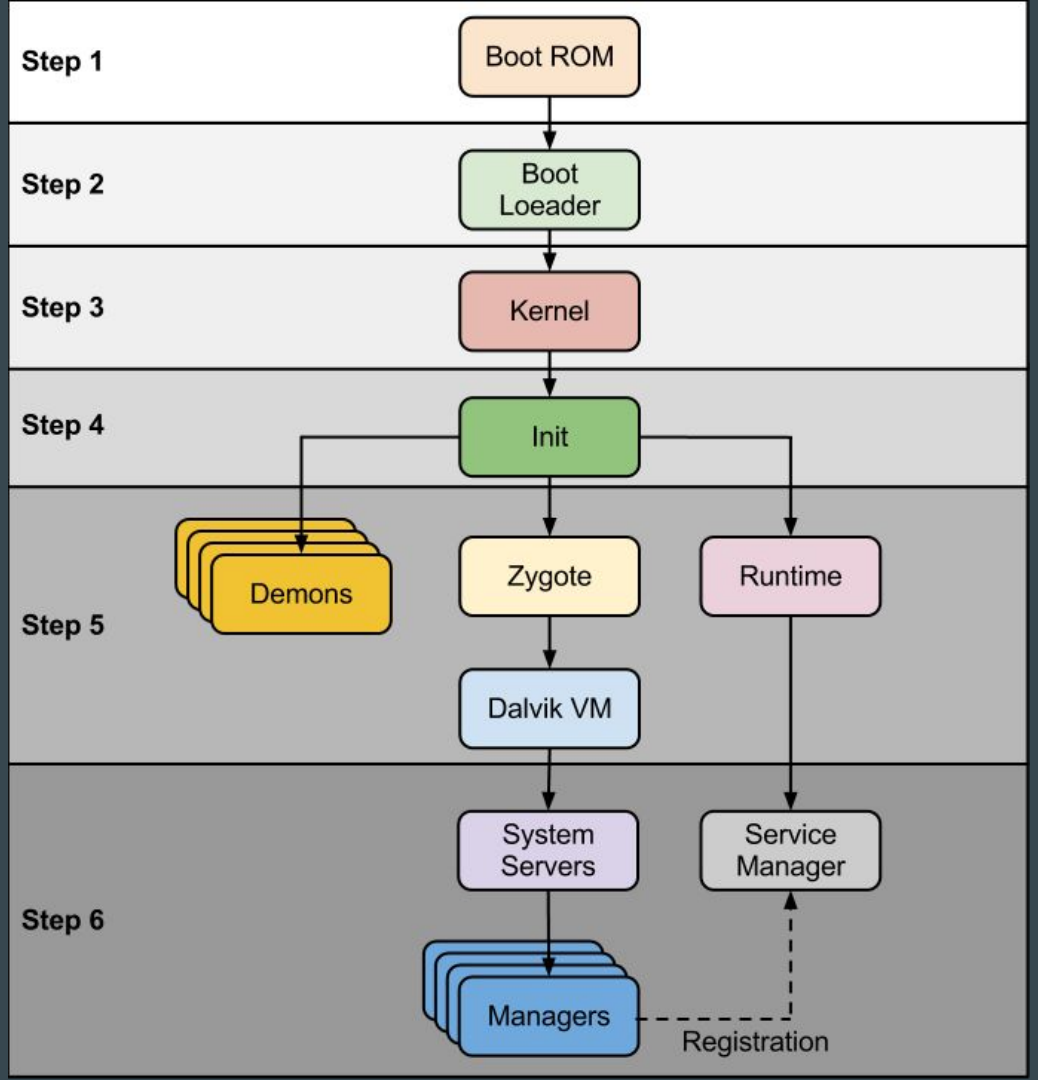
Android Security Concepts

- Permissions
  - Android 6.0 introduced dynamic permissions for certain messages
    - 7.5% of users have Android M [1]
  - Sandboxing enforced by UID
  - (each application is a different Linux user)
- Intents
  - Async messages passed between applications requesting data or to start an activity
- Built on Linux
  - SELinux, file permissions, system calls

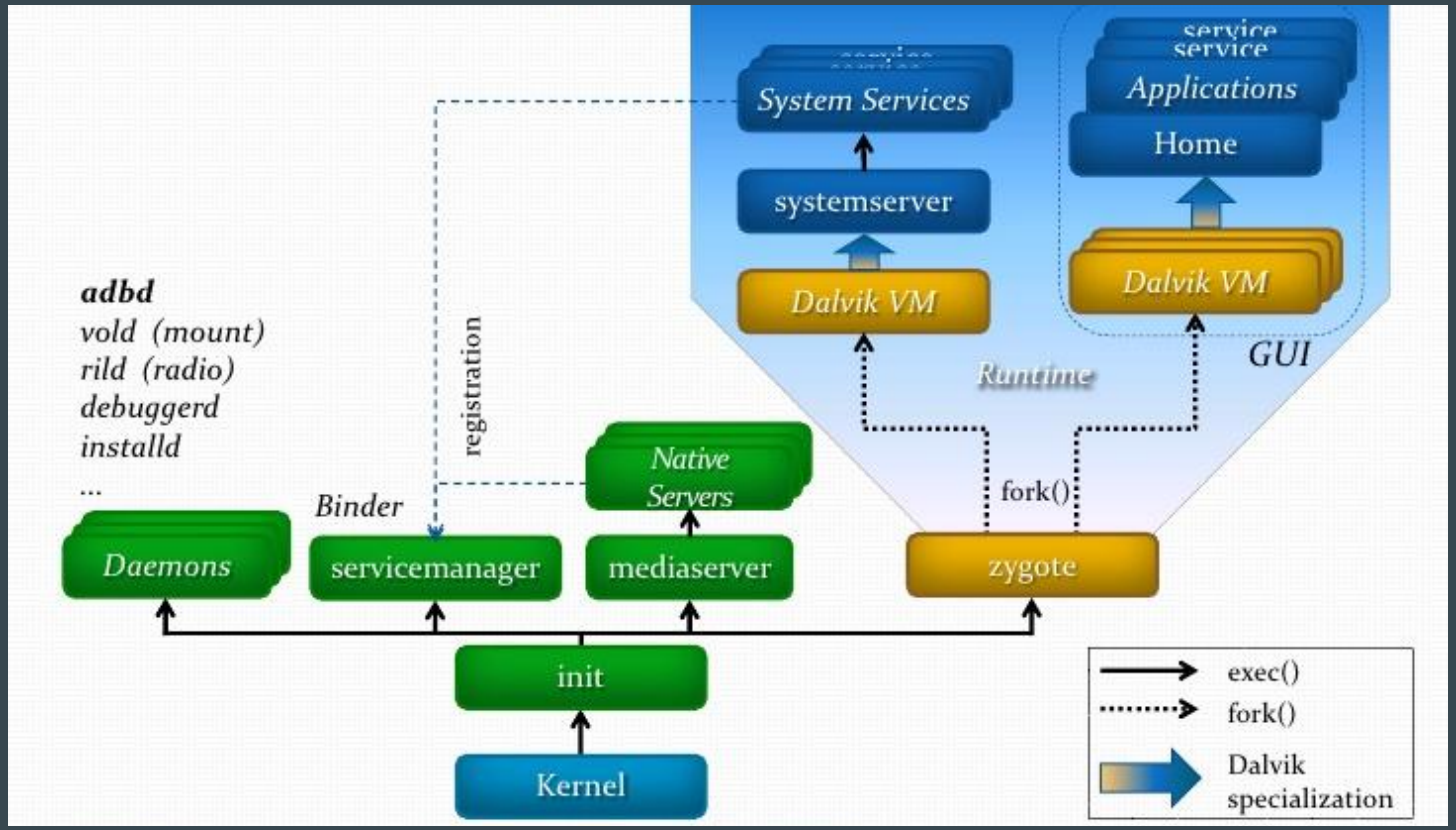




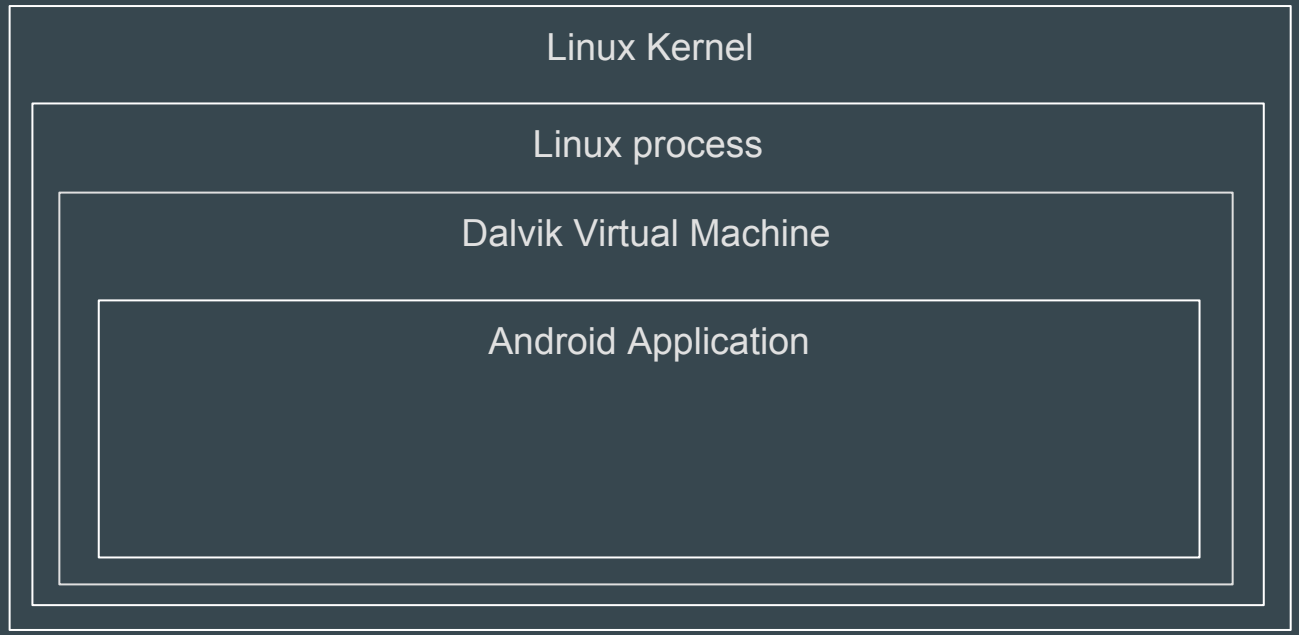
https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Android-System-Architecture.svg/2000px-Android-System-Architecture.svg.png



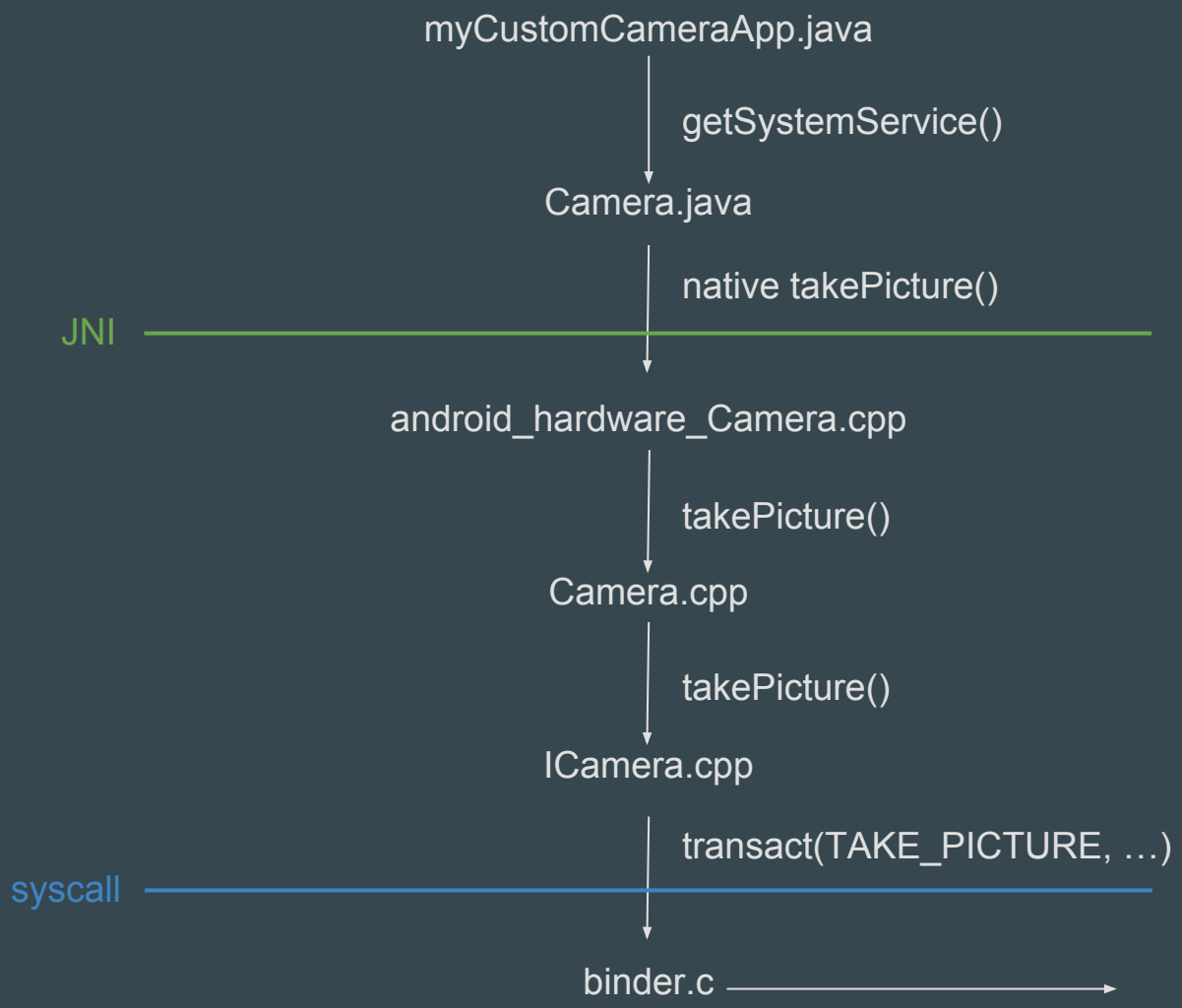
http://4.bp.blogspot.com/-uT2NBaV8WG8/UJuO0syJhnI/AAAAAAADgI/0CkrBvjyNDY/s1600/Android+Boot+Squence.png



http://image.slidesharecdn.com/jlstomoyotutorial-091023181710-phpapp02/95/learning-analyzing-and-protecting-android-with-tomoyo-linux-jls2009-10-728.jpg?cb=1256436625

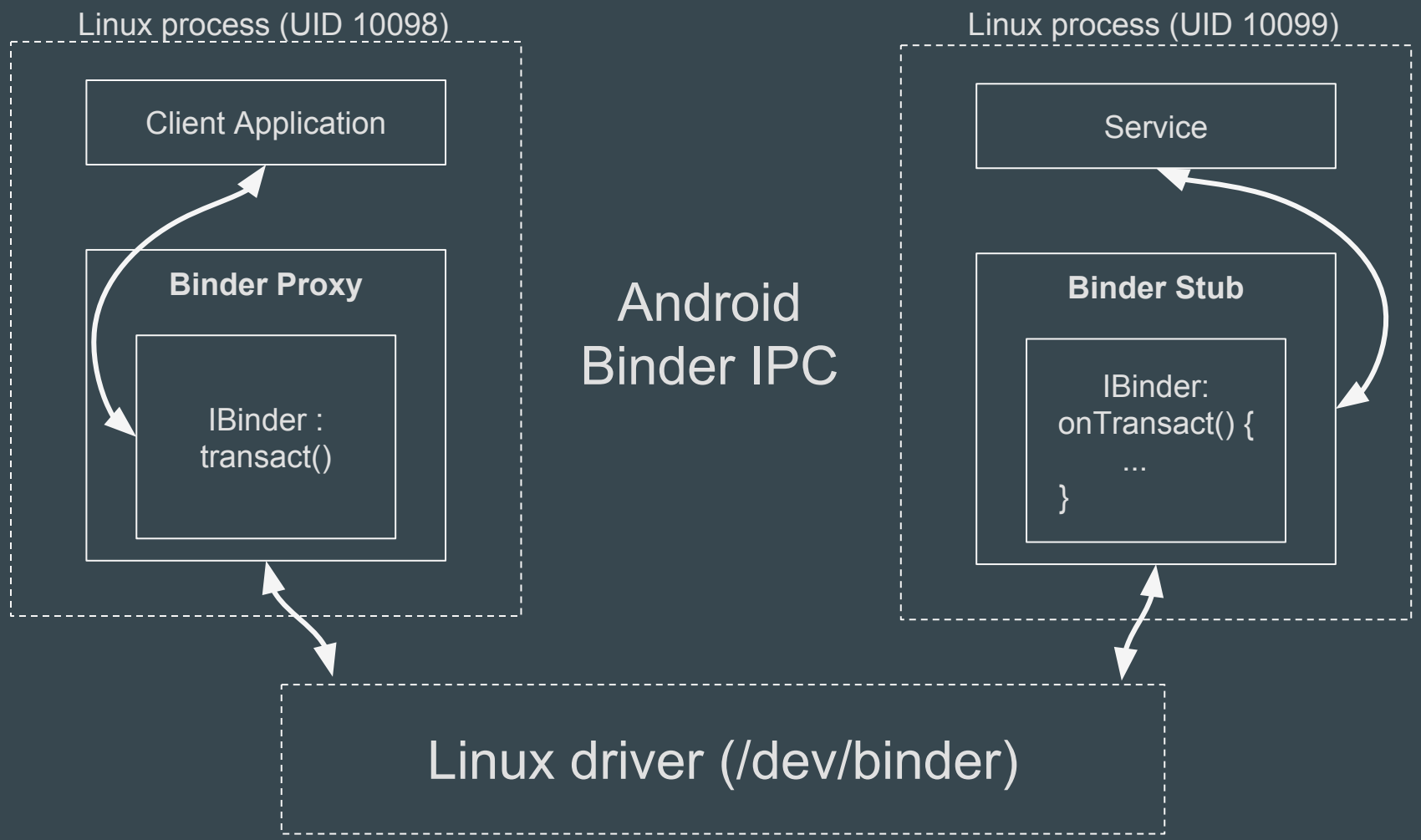


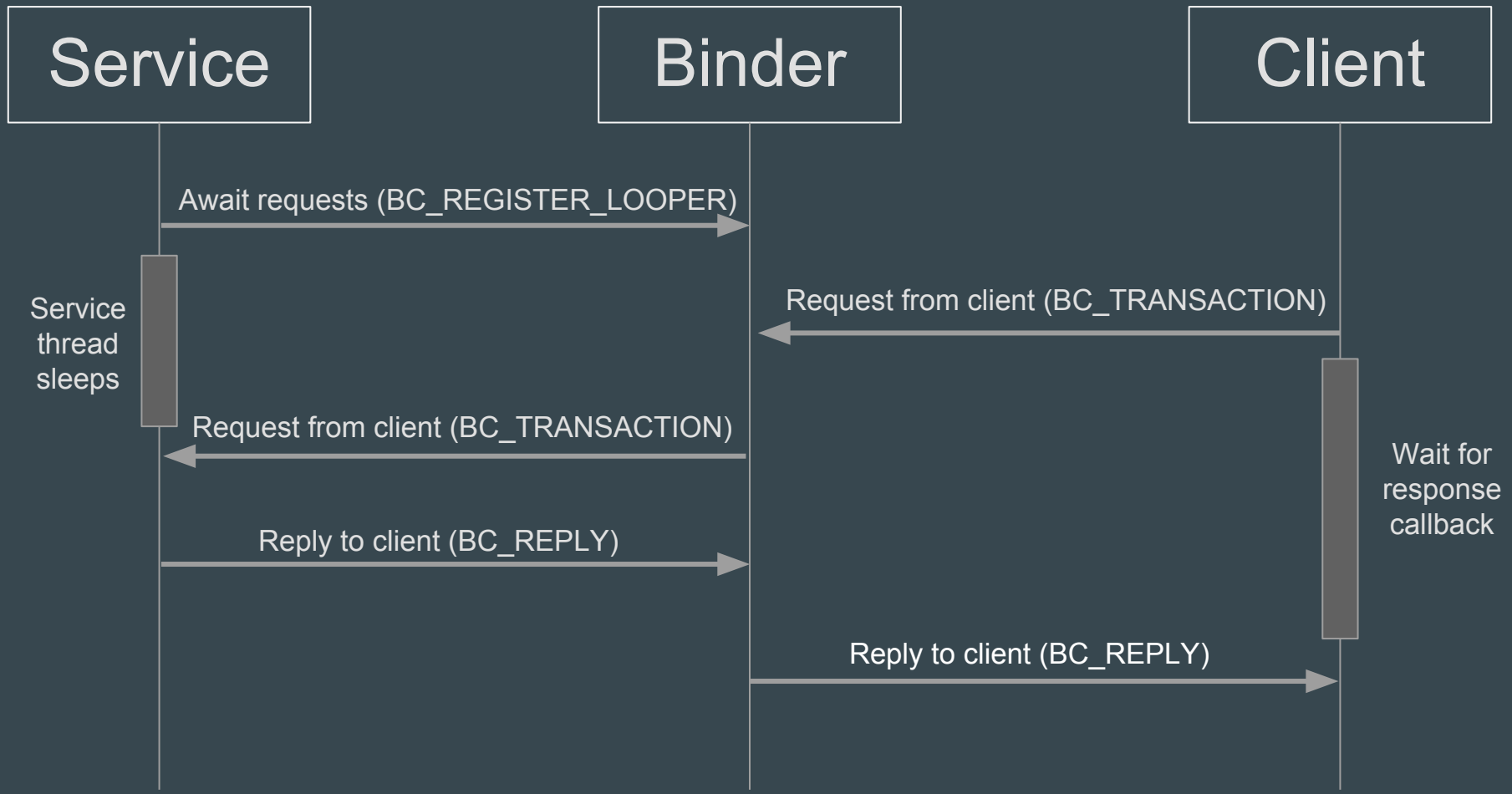
https://flexguruin.files.wordpress.com/2010/09/android_dalvik_vm.gif

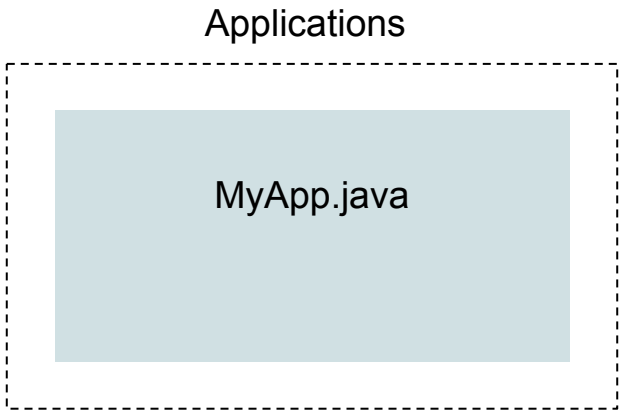

myCustomCameraApp.java
getSystemService()
Camera.java
native takePicture()
JNI
android_hardware_Camera.cpp
takePicture()
Camera.cpp
takePicture()
ICamera.cpp
transact(TAKE_PICTURE, …)
syscall
binder.c

Binder

- Android's IPC system (Linux IPC wasn't good enough)
- Supports tokens, death notifications, (local) RPC
- Every inter-application message (intent) goes through Binder
- Enables a client-server architecture with applications
- Implemented as a linux kernel driver (/dev/binder)
  - /drivers/staging/android/binder.{c,h}
- Userland applications call into the driver using ioctl()
- Binder driver copies data from process A to process B
- Intents, Messengers, and ContentProviders are built on Binder


Linux process (UID 10098)
Client Application
Binder Proxy
IBinder :
transact()
Android
Binder IPC
Linux process (UID 10099)
Service
Binder Stub
IBinder:
onTransact() {
...
}
Linux driver (/dev/binder)


Service
Binder
Client
Await requests (BC_REGISTER_LOOPER)
Service thread sleeps
Request from client (BC_TRANSACTION)
Request from client (BC_TRANSACTION)
Wait for response callback
Reply to client (BC_REPLY)
Reply to client (BC_REPLY)




```
Intent batteryStatus = Context.
registerReceiver(null, new
IntentFilter(      Intent.
ACTION_BATTERY_CHANGED);
```

Application Framework

ContextImpl.java

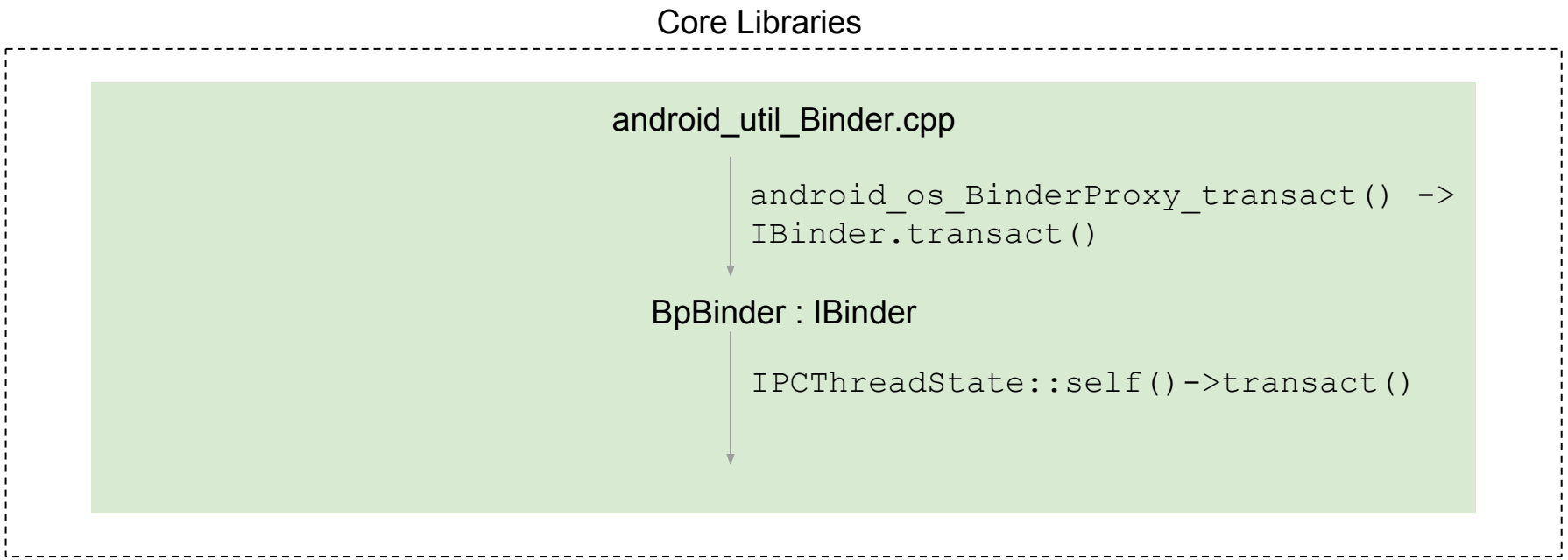
```
registerReceiver() ->
registerReceiverInternal()->
ActivityManagerNative.registerReceiver
()
```

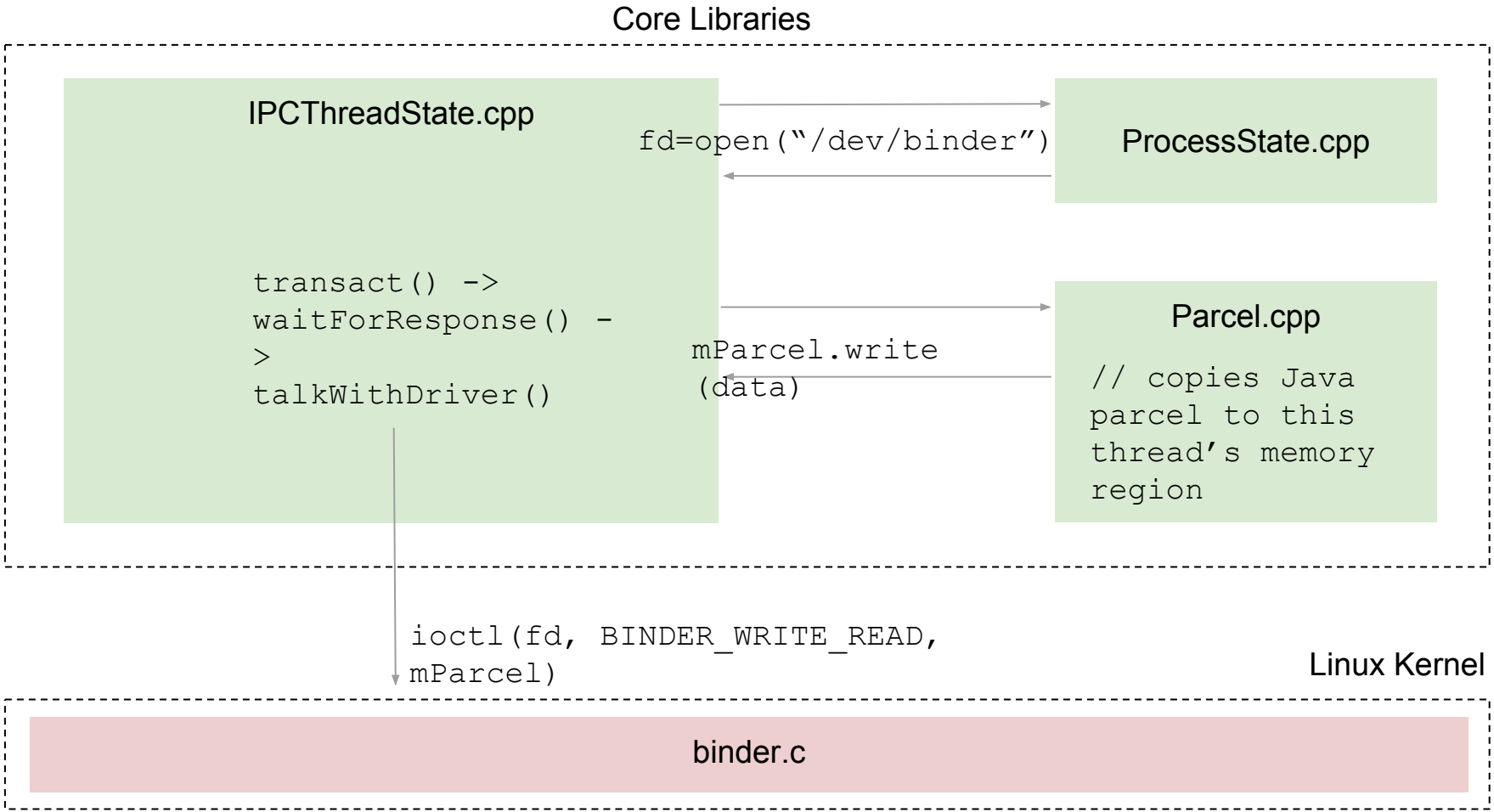
ActivityManagerNative.java

```
Parcel data = Parcel.obtain()
data.writeString(packageName)
filter.writeToParcel(data)
IBinder.transact(data, reply)
```

BinderProxy.java (implements IBinder)

```
transact() ->
native transactNative() //JNI
```


Core Libraries
android_util_Binder.cpp
android_os_BinderProxy_transact() ->
IBinder.transact()
BpBinder : IBinder
IPCThreadState::self()->transact()

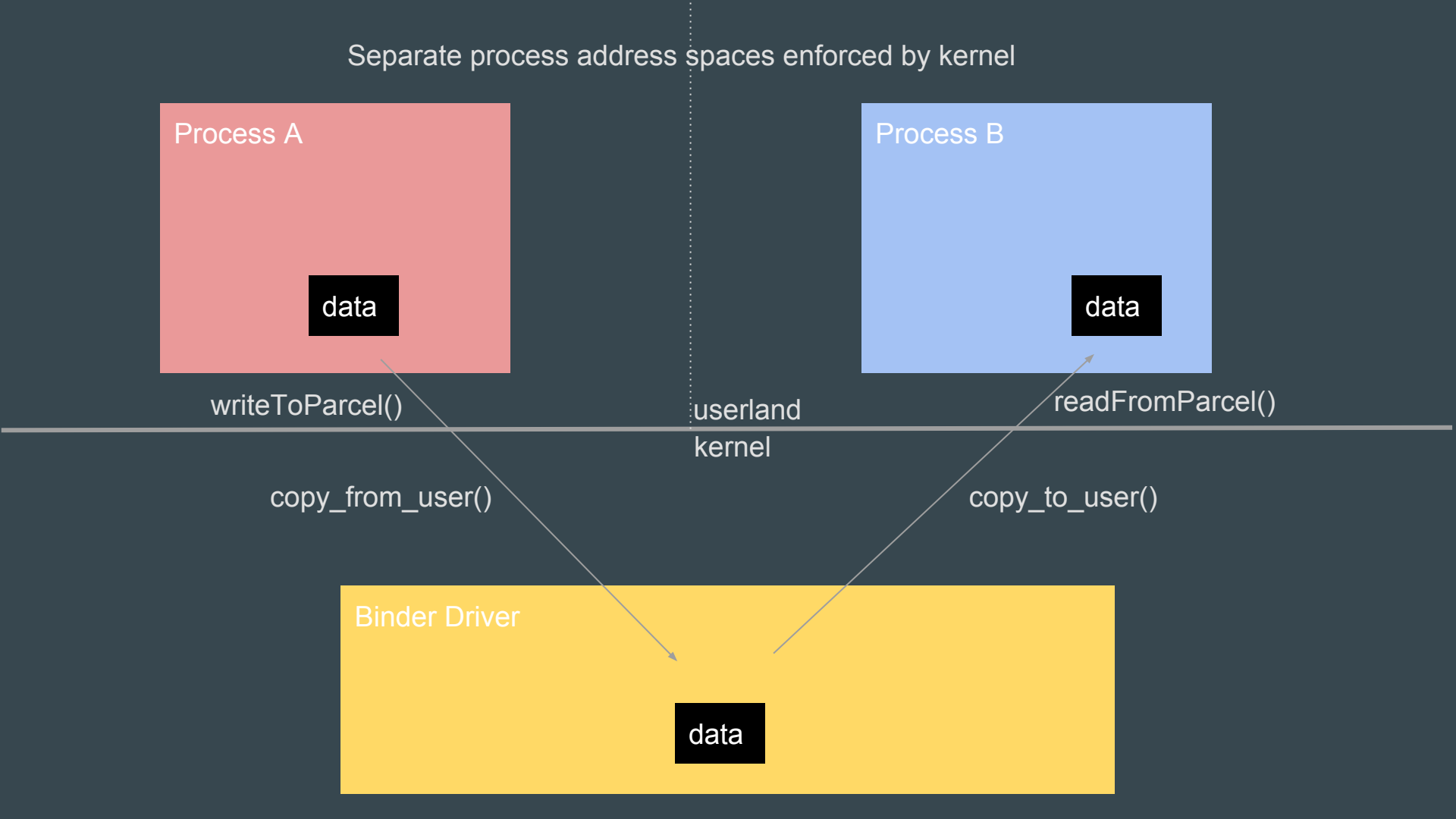

Core Libraries
IPCThreadState.cpp
fd=open("/dev/binder")
ProcessState.cpp
transact() -> waitForResponse() -> talkWithDriver()
mParcel.write (data)
Parcel.cpp
// copies Java parcel to this thread's memory region
ioctl(fd, BINDER_WRITE_READ, mParcel)
Linux Kernel
binder.c

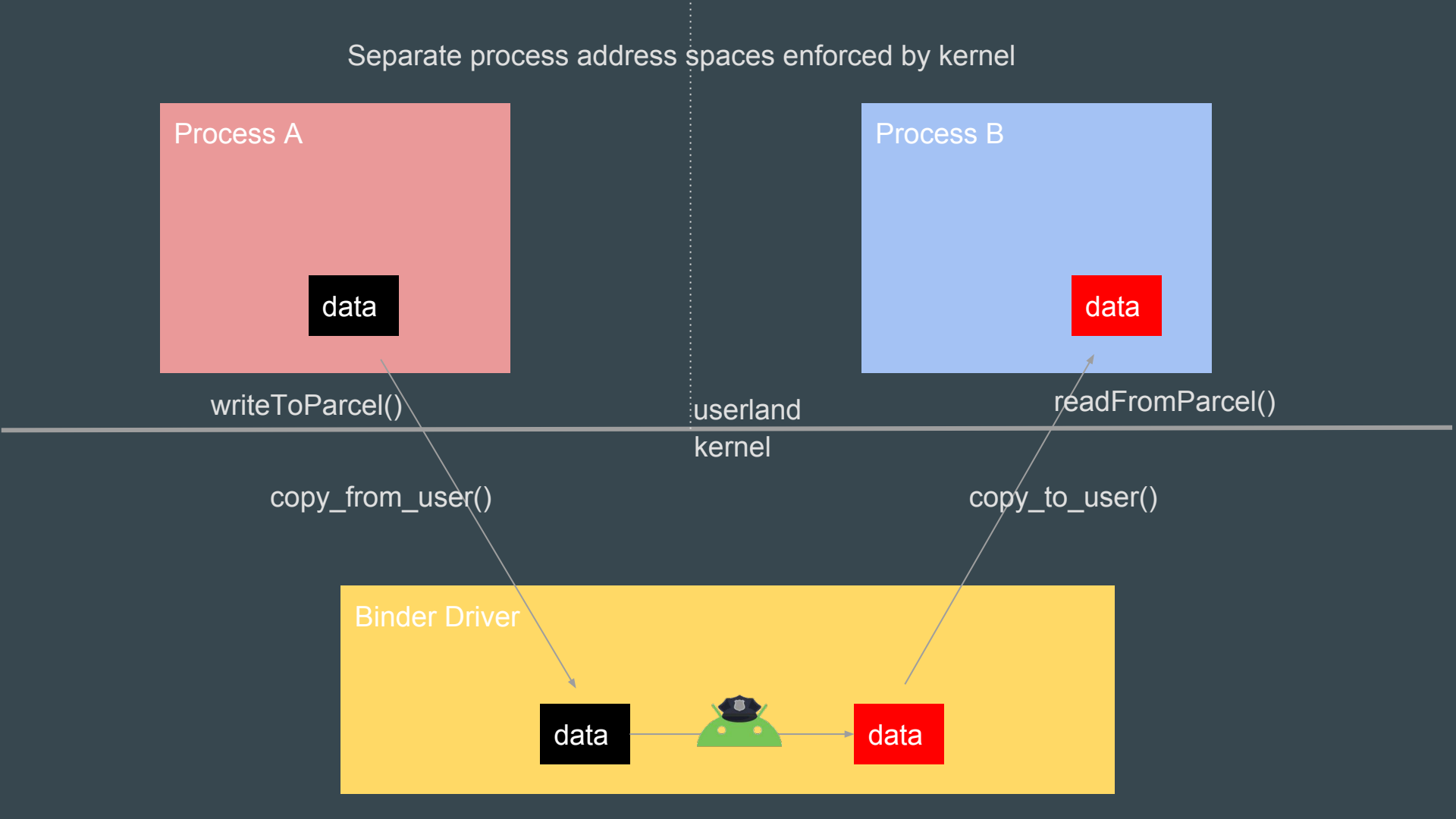
```
struct binder_transaction_data {
      /* The first two are only used for
      bcTRANSACTION and brTRANSACTION, identifying
      the target and contents of the transaction.
       */
      union {
            size_thandle;
            void  *ptr;
      } target;

      void        *cookie;
      unsigned intcode;
      unsigned intflags;

    /* General information about the transaction. */
      pid_t       sender_pid;
      uid_t       sender_euid;
      size_t      data_size;
      size_t      offsets_size;

      union {
            struct {
                  /* transaction data */
                  const void  *buffer;
                  const void  *offsets;
            } ptr;
            uint8_t     buf[8];
      } data;
};
```

```
struct binder_write_read {
      signed long write_size;
      signed long write_consumed;
      unsigned long write_buffer;
      signed long read_size;
      signed long read_consumed;
      unsigned long read_buffer;
};

struct flat_binder_object {
      /* 8 bytes for large_flat_header. */
      unsigned long type;
      unsigned long flags;

      /* 8 bytes of data. */
      union {
            void  *binder; // local obj
            signed long handle; // remote obj
      };

      /* extra data associated with local object */
      void *cookie;
};
```

binder.c (kernel driver)

1. device_initcall(binder_init); // called when kernel boots
2. binder_init()
   a. misc_register(&binder_miscdev) // register driver name and file operations
3. binder_ioctl() // entry point from userland
   a. wait_event_interruptable() // block caller until a response
   b. copy_from_user() // copy struct binder_write_read from userland
   c. binder_thread_write() or binder_thread_read() // depends on client or server request
4. binder_thread_write() // Called by client making a request
   a. Checks userland command // i.e. BC_TRANSACTION
   b. binder_transaction()
   c. copy_from_user(data) // copy struct binder_transaction_data from userland (buffer contents)
   d. list_add_tail(data, target) // add work to the target thread's queue
   e. wake_up_interruptable(target) // wake up the sleeping server thread
5. binder_thread_read() // Called by service thread waiting to handle requests
   a. while (1) { if (BINDER_LOOPER_NEED_DATA) goto retry; }
   b. data = list_first_entry() // get request data
   c. copy_to_user(data) // copy the data to service


Separate process address spaces enforced by kernel
Process A
data
Process B
data
writeToParcel()
readFromParcel()
userland
kernel
copy_from_user()
copy_to_user()
Binder Driver
data


Separate process address spaces enforced by kernel
Process A
data
Process B
data
writeToParcel()
readFromParcel()
userland
kernel
copy_from_user()
copy_to_user()
Binder Driver
data
data

Binder hook

http://androidxref.com/kernel_3.18/xref/drivers/staging/android/binder.c#1520

```
#include "binder_filter.h"
extern int filter_binder_message(unsigned long, signed long, int, int, void*, size_t);

…
static void binder_transaction(struct binder_proc *proc, struct binder_thread *thread,
                               struct binder_transaction_data *tr, int reply)
 {
     struct binder_transaction *t = kzalloc(sizeof(*t), GFP_KERNEL);
     …
     if (copy_from_user(t->buffer->data, tr->data.ptr.buffer, tr->data_size)) {
          ...
          goto err_copy_data_failed;
     }
     …
     filter_binder_message((unsigned long)(t->buffer->data), tr->data_size, reply,
                           t->sender_euid, (void*)offp, tr->offsets_size);
     …
}
```

Benefits of being in the kernel binder driver

- Alternative: Userland Android library hooks (or Xposed framework hooks)
  - Safer, quicker code
- Complete mediation
  - Direct binder messages are possible (app to app, app to service)
  - ServiceManager is not in a position for complete mediation: apps can register binder receivers that don't go through ServiceManager [4]
    - Intent.registerReceiverAsUser()
    - Service.bindService()
  - Get context data directly from the System sensors (based on UID)

Logging

- Existing binder.c logs
  a. printk(), TRACE_EVENT(), seq_printf()
- Existing: [49431.544219] binder: 9916:9916 BC_TRANSACTION 683674 -> 198 - node 289403, data 8dc12180 (null) size 80-0
- Pretty print: [14:33:56.084452] binder_command BC_TRANSACTION: process pid 9916 (android.picky), thread pid 9916 -> process pid 198 (/system/bin/surfaceflinger), node id 289403, transaction id 683674, data address 8dc12180, data size 80, offsets address null, offsets size 0

Logging (cont.)

Binder buffer contents in memory

- Contents printed to kernel debug buffer when module parameter is set

```
{(0)(64)(24)(0)android.os.IPowerManager(0)(0)(1)(0)(0)(0)}
```

Each character or (p) value is a byte

Strings are prepended by their length

Classes are passed as string literals

Sender of the intent

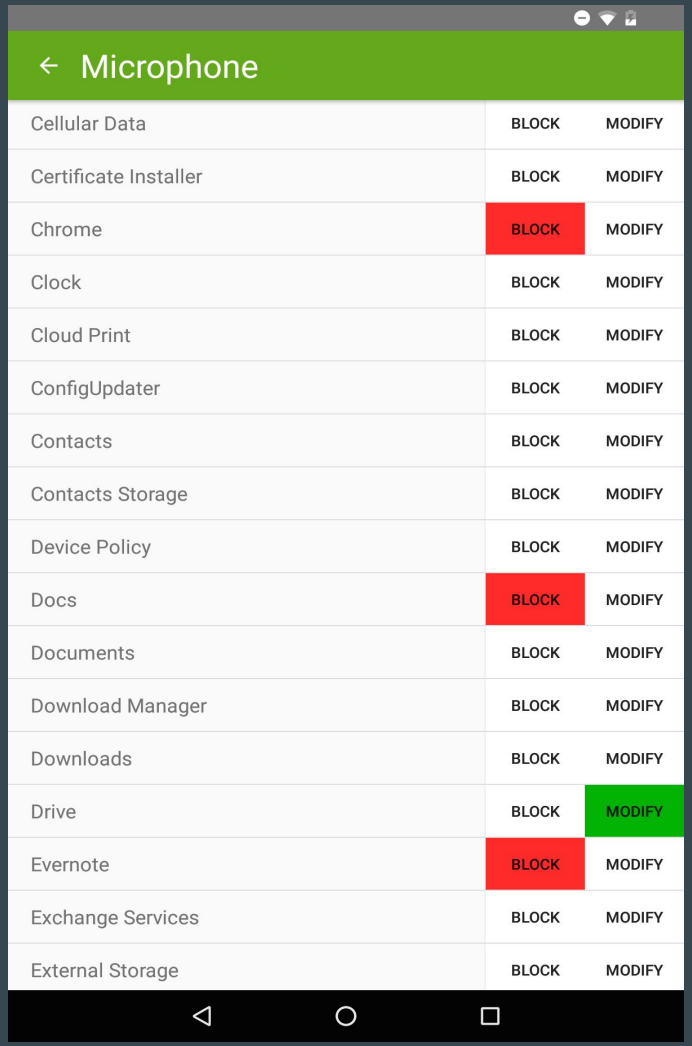
Chars are 2 bytes

Integers are 4 bytes

Fields are aligned on 4 byte intervals

Picky

- Android application
- Allows user to dynamically set policy
- Import/Export policy
- Policy persists across app sessions and reboot
- Requires lockscreen
- Specify context, custom messages
- Per-app blocking



Picky-BinderFilter interface

- Android's NDK (Native Development Kit) allows Java apps to call Native C++ code through the JNI (Java Native Interface) framework [8]
- We use this to call `sys_open`, `copy_to_user`/`sys_read`, and `copy_from_user`/`sys_write` to read and write userland policy to/from the BinderFilter kernel driver

Demo

Blocking messages

Blocking messages

- Blocking generic strings in Binder messages is dangerously powerful
- Check uid, context
- Check binder message content for message with `strstr`
- Clear out the entire message with `memset`

Intents

- Ways to get around using intents
  - Camera: applications like VSCO (fancy camera app) and GoogleCamera implement their own camera, call Android Camera API [5, 6]
- Permissions encapsulate multiple intents with the granularity users can understand
  - Location: multiple ways to get phone location means multiple intents to block [7]. We can block all of them with one permission!

Blocking Permissions

```
{(0)@(28)(0)android.app.IActivityManager(0)(0))(0)android.
permission.ACCESS_COARSE_LOCATION(0)(155)(9)(0))'(0)}
```

note: uid is passed in as hook function parameter

Demo

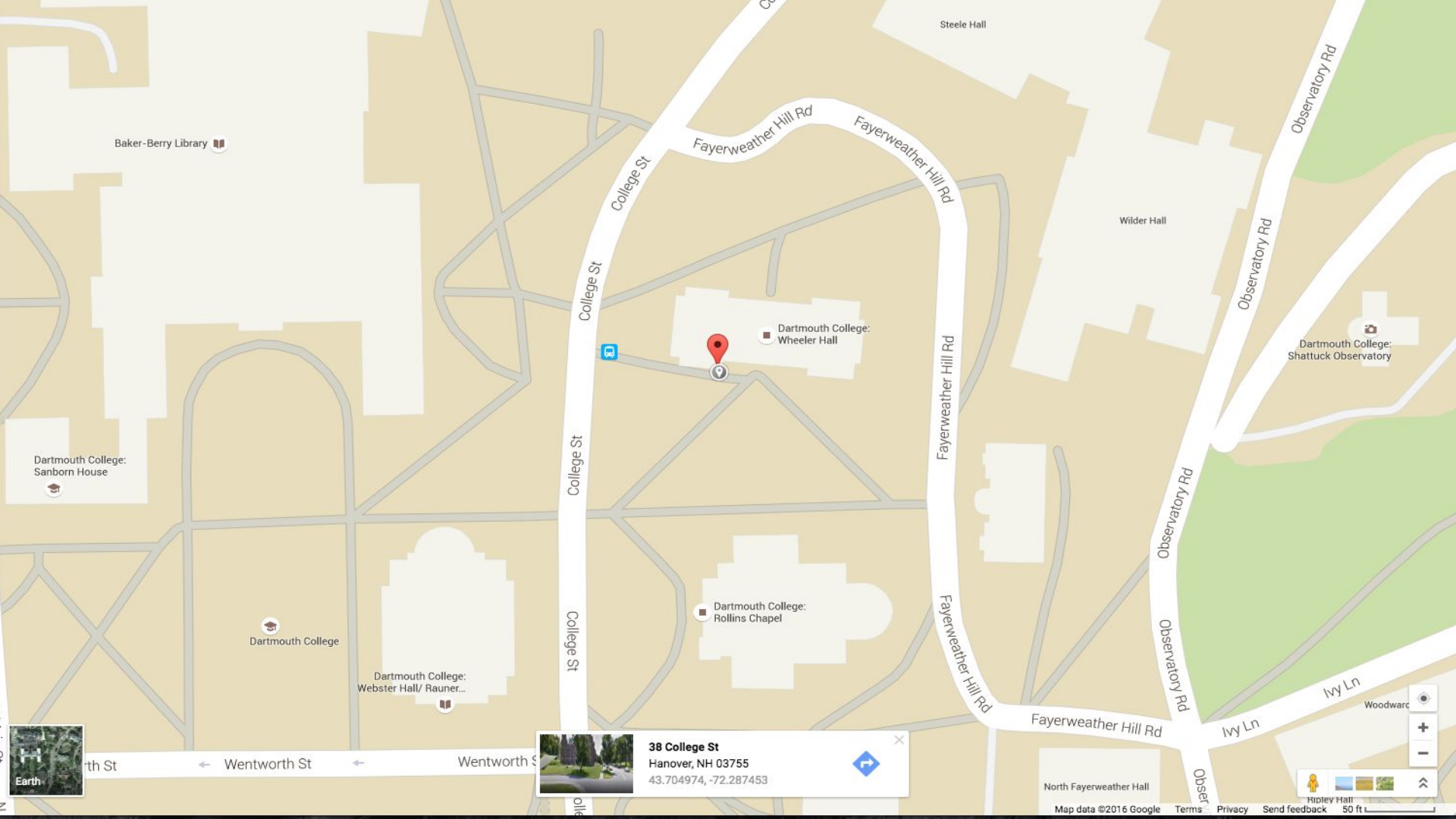
Context

Wifi SSID context

```
{(0)@(30)(0)android.app.IApplicationThread(0)(0)(133)h(127)
(07)(247)(07)(07)(07)$(07)android.net.conn.CONNECTIVITY_CHANGE
(07)(07)(07)(07)(255)(255)(16)(0)(255)(255)(255)(255)(05)(05)
(05)(05)(05)(05)(05)(05)(254)(255)x(05)BD(45)(05)(11)(0)
networkInfo(0)(4)(0)(23)(0)android.net.NetworkInfo(0)(1)(0)(0)
(0)(4)(0)WIFI(0)(0)(0)(0)(0)(0)(9)(0)CONNECTED(0)(9)(0)
CONNECTED (0)(0)(0)(1)(0)(0)(0)(255)(255)(18)(0)"SecureNet"(0)
(0)(11)(0)networkType(0)(1)(0)(1)(0)(13)(0)inetCond}
```

GPS location context

```
{(0)@(29)(0)android.content.IIntentSender(0)(0)(0)(1)(0)
(255)(255)(255)(255)(0)(0)(255)(255)(255)(255)(0)(0)(255)
(255)(255)(255)(255)(255)(255)(255)(0)(0)(0)(0)(0)(0)(0)(0)
(254)(255)(255)(255)(224)(4)(0)BNDL(3)(0)8(0)com.google.
android.location.internal.EXTRA_LOCATION_LIST(0)(0)(11)(0)
(1)(0)(4)(0)
(25)(0)android.location.Location(0)(7)(0)network(0)
(192)(191)(187)(145)T(1)(0)@(165)R(132)\(0)(177)(237)(254)
(194)(60)(218)(69)(64)(121)(189)(234)(183)(101)(18)(82)(192)
(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(1)(0)u(19)(}

*(double*)({177,237,254,194,60,218,69,64}) = 43.704979
*(double*)({121,189,234,183,101,18,82,192}) = -72.287458
```


Steele Hall
Baker-Berry Library
Fayerweather Hill Rd
Fayerweather Hill Rd
College St
Wilder Hall
Observatory Rd
Observatory Rd
Dartmouth College: Wheeler Hall
Dartmouth College: Shattuck Observatory
Dartmouth College: Sanborn House
College St
Fayerweather Hill Rd
Observatory Rd
Dartmouth College: Rollins Chapel
Dartmouth College
Dartmouth College: Webster Hall/ Rauner...
College St
Fayerweather Hill Rd
Observatory Rd
Ivy Ln
Ivy Ln
Fayerweather Hill Rd
Wentworth St
Wentworth St
Wentworth S
Earth
38 College St
Hanover, NH 03755
43.704974, -72.287453
North Fayerweather Hall
Ripley Hall
Map data ©2016 Google
Terms
Privacy
Send feedback
50 ft

Demo

Modifying pictures

Modifying Saved Pictures

{(4)H(28)(0)android.app.IActivityManager(0)(0)(133)*bs(127)
(1)(0)P(196)(180)(174)(224)(145)(181)(172)(19)(0)com.
facebook.katana(0)"(0)android.media.action.IMAGE_CAPTURE(0)
(0)(0)(0)(255)(255)(255)(255)(3)(0)(255)(255)(255)(255)(255)
(255)(255)(255)(0)(0)(0)(0)(0)(0)(1)(0)(1)(0)(0)(0)(0)(0)(1)
(0)(13)(0)text/uri-list(0)(0)(0)(1)(0)(1)(0)(255)(255)(255)
(255)(255)(255)(255)(255)(0)(0)(1)(0)(3)(0)(4)(0)file(0)(0)
(0)(0)(0)(0)(0)(0)(0)(0)(0)(0)(2)(0)(62)(0)
/storage/emulated/0/Pictures/Facebook/FB_IMG_1464314001208.
jpg}

Modifying Saved Pictures (cont.)

```
static void copy_file_to_file(char* filename_src, char* filename_dst)
{
     ...
     set_fs(KERNEL_DS);    // sys_open expects USERLAND addresses: trick it
     fd_read = sys_open(filename_src, O_RDONLY, 0);
     fd_write = sys_open(filename_dst, O_WRONLY|O_CREAT|O_TRUNC, 0644);
     ...
     while (1) {
          read_len = sys_read(fd_read, read_buf, buf_size-1);
          if (read_len <= 0) {
               break;
          }
          sys_write(fd_write, read_buf, read_len);
          write_file = fget(fd_write);
          ...
          vfs_write(write_file, read_buf, read_len, &pos);
          fput(write_file);
     }
     ...
}
```

Blocking system permissions

- Some permissions are “system only” [2]
  - Applications that are located in /system/app and /system/priv-app, ex. Chrome, Settings
- PackageManager checks system applications’ permissions differently
- PackageManagerService.installPackageAsUser() [3]

```
// Only system components can circumvent runtime permissions when installing.
if ((installFlags & PackageManager.INSTALL_GRANT_RUNTIME_PERMISSIONS) != 0
        && mContext.checkCallingOrSelfPermission(Manifest.permission
        .INSTALL_GRANT_RUNTIME_PERMISSIONS) == PackageManager.
PERMISSION_DENIED) {
    throw new SecurityException("You need the "
            + "android.permission.INSTALL_GRANT_RUNTIME_PERMISSIONS permission
"
            + "to use PackageManager.INSTALL_GRANT_RUNTIME_PERMISSIONS flag");
}
```

Demo

Blocking system permissions

Blocking system permissions (cont.)

Sent before Google Play Store installs the “com.groupme.android” package:

```
{(0)(64)(28)(0)android.app.IActivityManager(0)(0)(1)(0)(19)
(0)com.android.vending(0)(133)*bs(127)(1)(0)(0)(0)(0)(0)
(255)(255)(255)(255)k(157)+m(1)(0)(1)(0)(1)(0)E(0)com.
android.vending.INTENT_PACKAGE_INSTALL_COMMIT.com.groupme.
android(0)(0)(0)(255)(255)(255)(255)(0)(0)(255)(255)(255)
(255)(255)(255)(255)(255)(0)(0)(0)(0)(0)(0)(0)(0)(254)(255)
(255)(255)(255)(255)(255)(255)(255)(255)(255)(255)(0)(0)H(0)
(0)(0)(0)}
```

Discussion & future work

- Discussion:
  - Blocking permissions dynamically crashes apps
    - Android recommends dynamically checking permissions
  - Default blocking for certain messages
- Future work:
  - More contexts
  - Message modification library
  - Customize SELinux policy settings in Android
  - Profiling performance overhead, static code analysis

References

[1] https://developer.android.com/about/dashboards/index.html

[2] https://developer.android.com/reference/android/Manifest.permission.html#BLUETOOTH_PRIVILEGED

[3] http://androidxref.com/6.0.1_r10/xref/frameworks/base/services/core/java/com/android/server/pm/PackageManagerService.java#9557

[4] https://developer.android.com/guide/components/bound-services.html#Creating

[5] http://androidxref.com/6.0.1_r10/xref/frameworks/base/core/java/android/hardware/Camera.java#1412

[6] https://developer.android.com/reference/android/hardware/Camera.html

[7] https://developer.android.com/guide/topics/location/strategies.html

[8] http://tools.android.com/tech-docs/android-ndk-preview

Questions