Beyond Java: Obfuscating Android Apps with Purely Native Code

Laurie Kirk

whoami

- Laurie Kirk
- Reverse Engineer at Microsoft
- Specialize in cross-platform malware with a focus on mobile malware
- Run YouTube channel @lauriewired
- Representing myself as an individual security researcher today (not representing Microsoft)

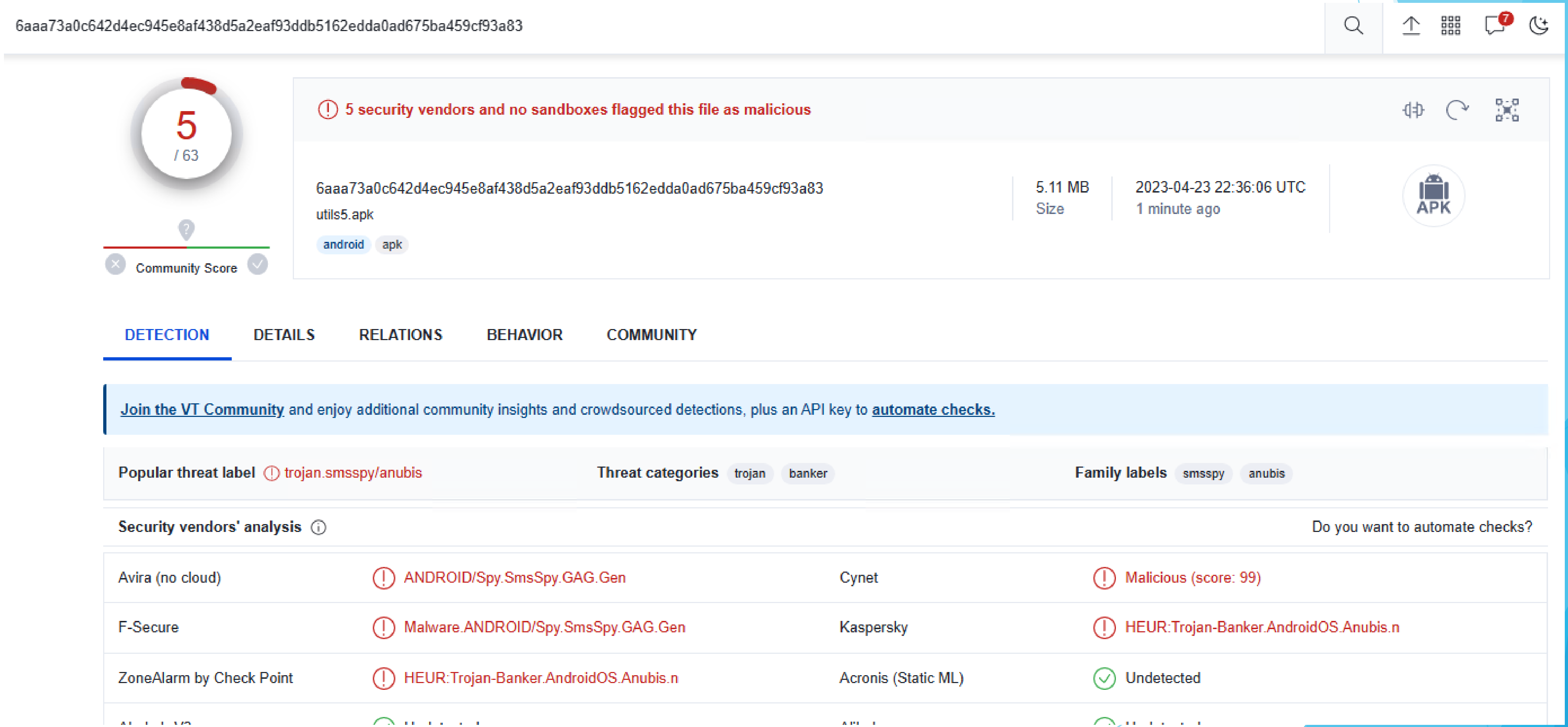
@lauriewired

Analysis Materials

LaurieWired TROOPERS23
Github Repo

https://github.com/LaurieWired/Android
PurelyNative_Troopers23

The only difference between this app...



6aaa73a0c642d4ec945e8af438d5a2eaf93ddb5162edda0ad675ba459cf93a83

5 / 63

Community Score

5 security vendors and no sandboxes flagged this file as malicious

6aaa73a0c642d4ec945e8af438d5a2eaf93ddb5162edda0ad675ba459cf93a83
utils5.apk
android apk

5.11 MB
Size

2023-04-23 22:36:06 UTC
1 minute ago

APK

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

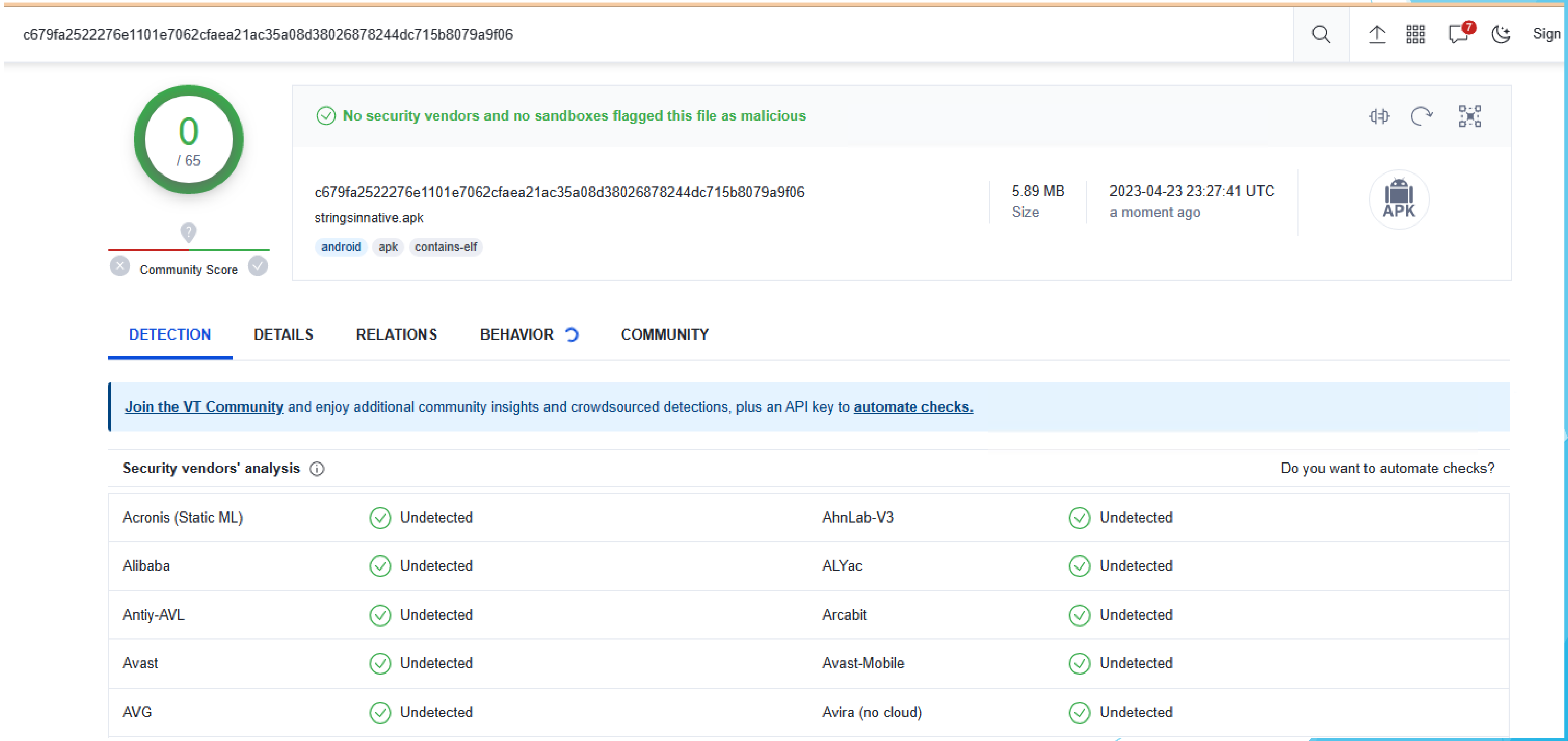
Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.smsspy/anubis Threat categories trojan banker Family labels smsspy anubis

Security vendors' analysis Do you want to automate checks?

Avira (no cloud)	ANDROID/Spy.SmsSpy.GAG.Gen	Cynet	Malicious (score: 99)
F-Secure	Malware.ANDROID/Spy.SmsSpy.GAG.Gen	Kaspersky	HEUR:Trojan-Banker.AndroidOS.Anubis.n
ZoneAlarm by Check Point	HEUR:Trojan-Banker.AndroidOS.Anubis.n	Acronis (Static ML)	Undetected

… and this app



c679fa2522276e1101e7062cfaea21ac35a08d38026878244dc715b8079a9f06

Sign

0
/ 65

Community Score

No security vendors and no sandboxes flagged this file as malicious

c679fa2522276e1101e7062cfaea21ac35a08d38026878244dc715b8079a9f06
stringsinnative.apk
android apk contains-elf

5.89 MB
Size

2023-04-23 23:27:41 UTC
a moment ago

APK

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected
Avast	Undetected	Avast-Mobile	Undetected
AVG	Undetected	Avira (no cloud)	Undetected

is that part of the code is written in C++.

Agenda

- Obfuscate an Android app
- Use purely native code
- Mask our API calls

Java is the main language in Android

Managed code

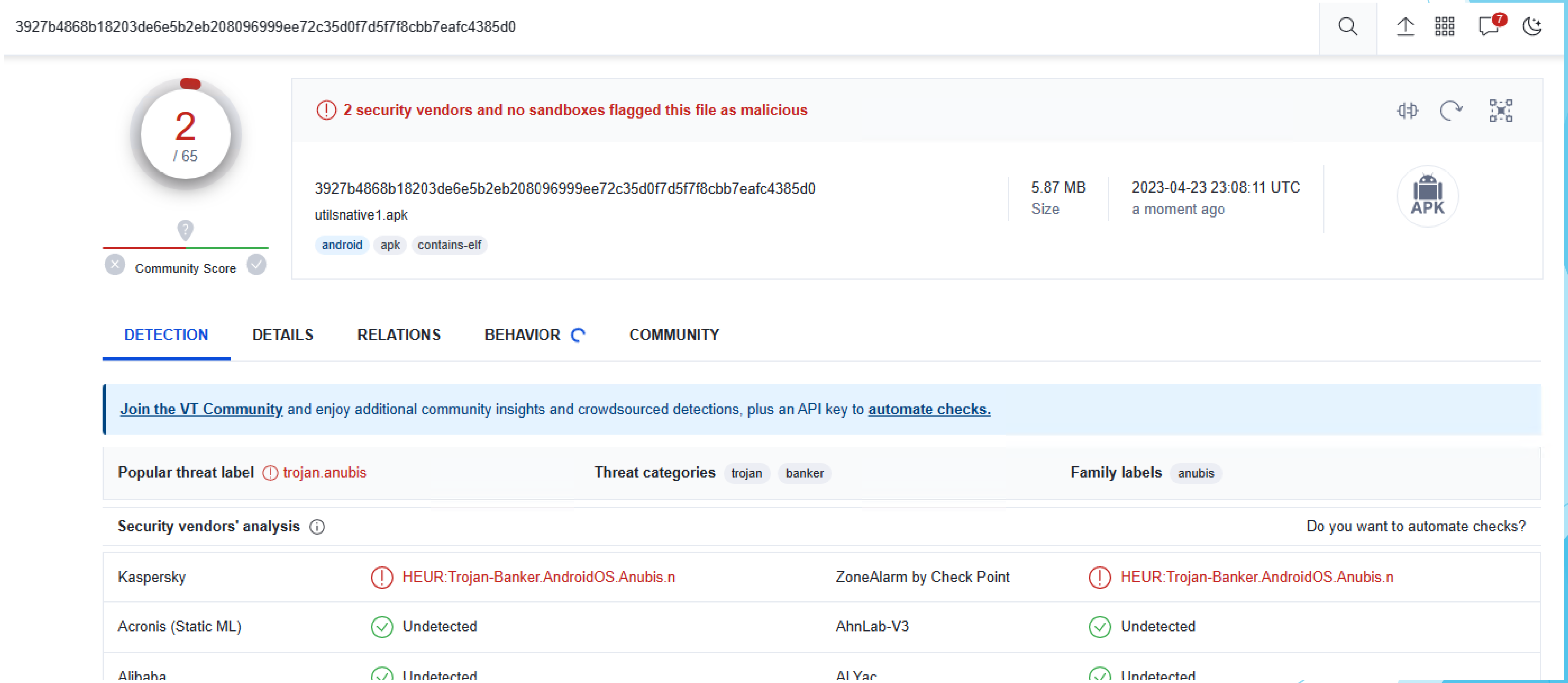
Java / Kotlin

Native code

C / C++

C++ == obfuscation?

Same file except I added a blank C++ stub



3927b4868b18203de6e5b2eb208096999ee72c35d0f7d5f7f8cbb7eafc4385d0

2 / 65

Community Score

2 security vendors and no sandboxes flagged this file as malicious

3927b4868b18203de6e5b2eb208096999ee72c35d0f7d5f7f8cbb7eafc4385d0
utilsnative1.apk
android apk contains-elf

5.87 MB
Size

2023-04-23 23:08:11 UTC
a moment ago

APK

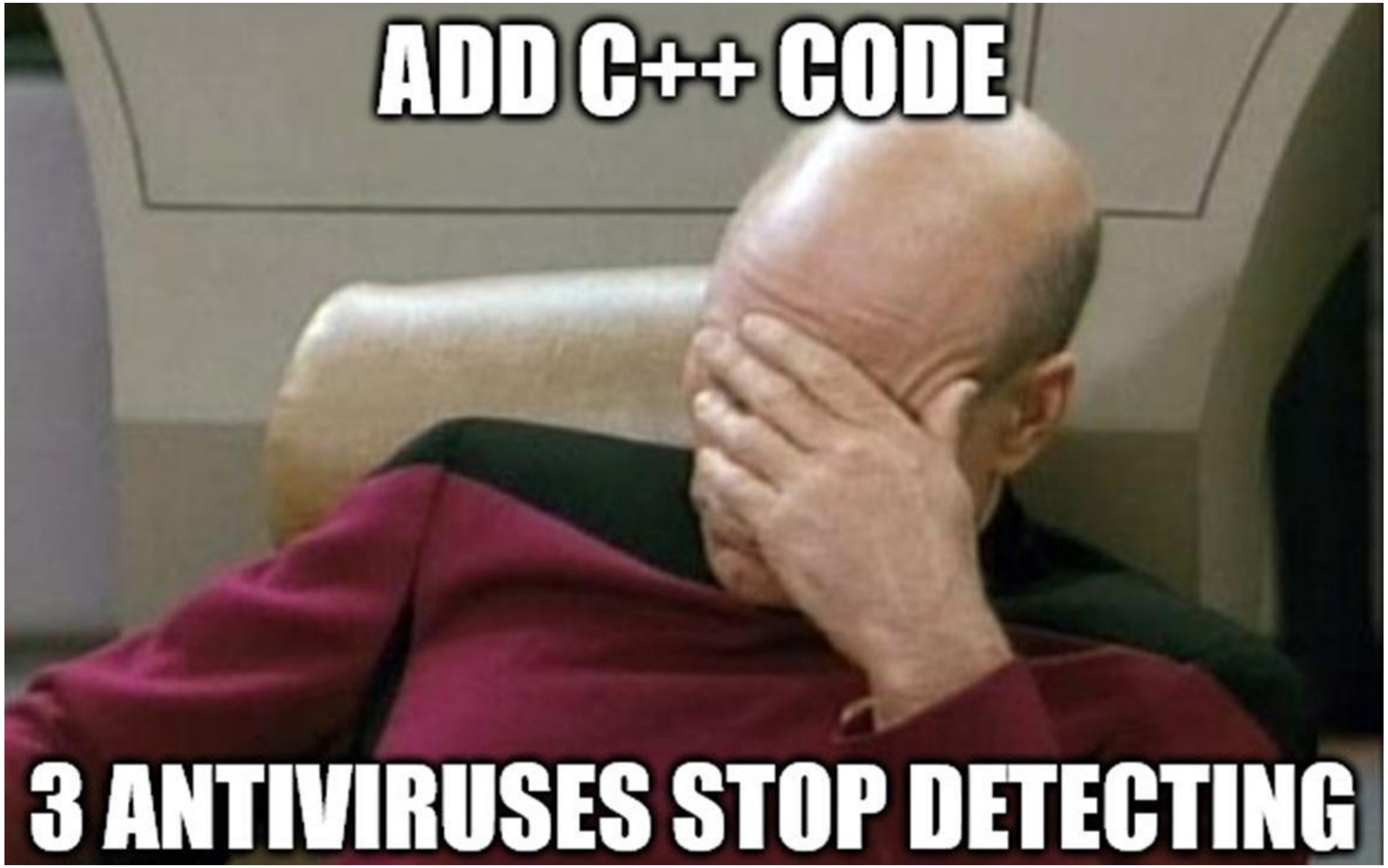
DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.anubis — Threat categories trojan banker — Family labels anubis

Security vendors' analysis — Do you want to automate checks?

Kaspersky	HEUR:Trojan-Banker.AndroidOS.Anubis.n	ZoneAlarm by Check Point	HEUR:Trojan-Banker.AndroidOS.Anubis.n
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected


ADD C++ CODE
3 ANTIVIRUSES STOP DETECTING

Further Native Obfuscation Advantages

- More challenging to reverse engineer
  - Read assembly instead of Java
  - Understand JNI invocations
- Remove x86 support to thwart emulators

How far can we go?

Purely Native Code Methodology

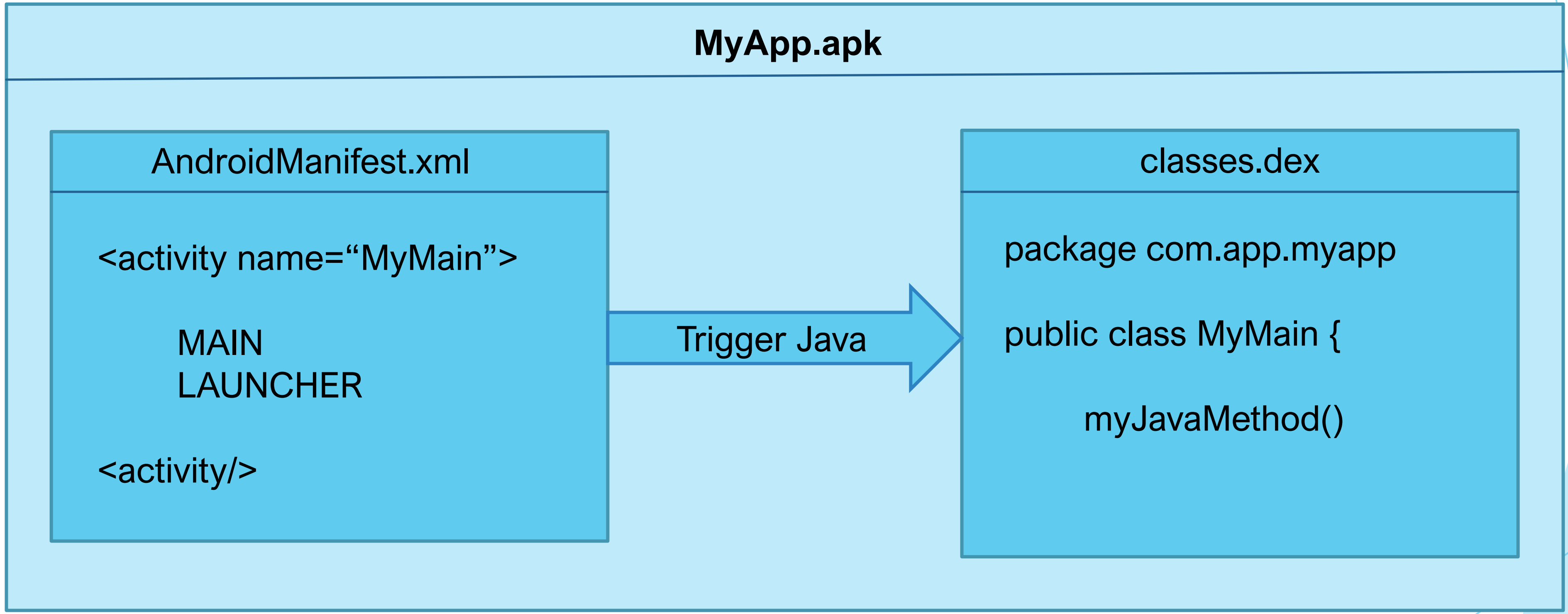
Remove Java entrypoint

Translate methods to C++

Conceal Android API calls

Removing the Java Entrypoint

The Manifest defines entrypoints in Java



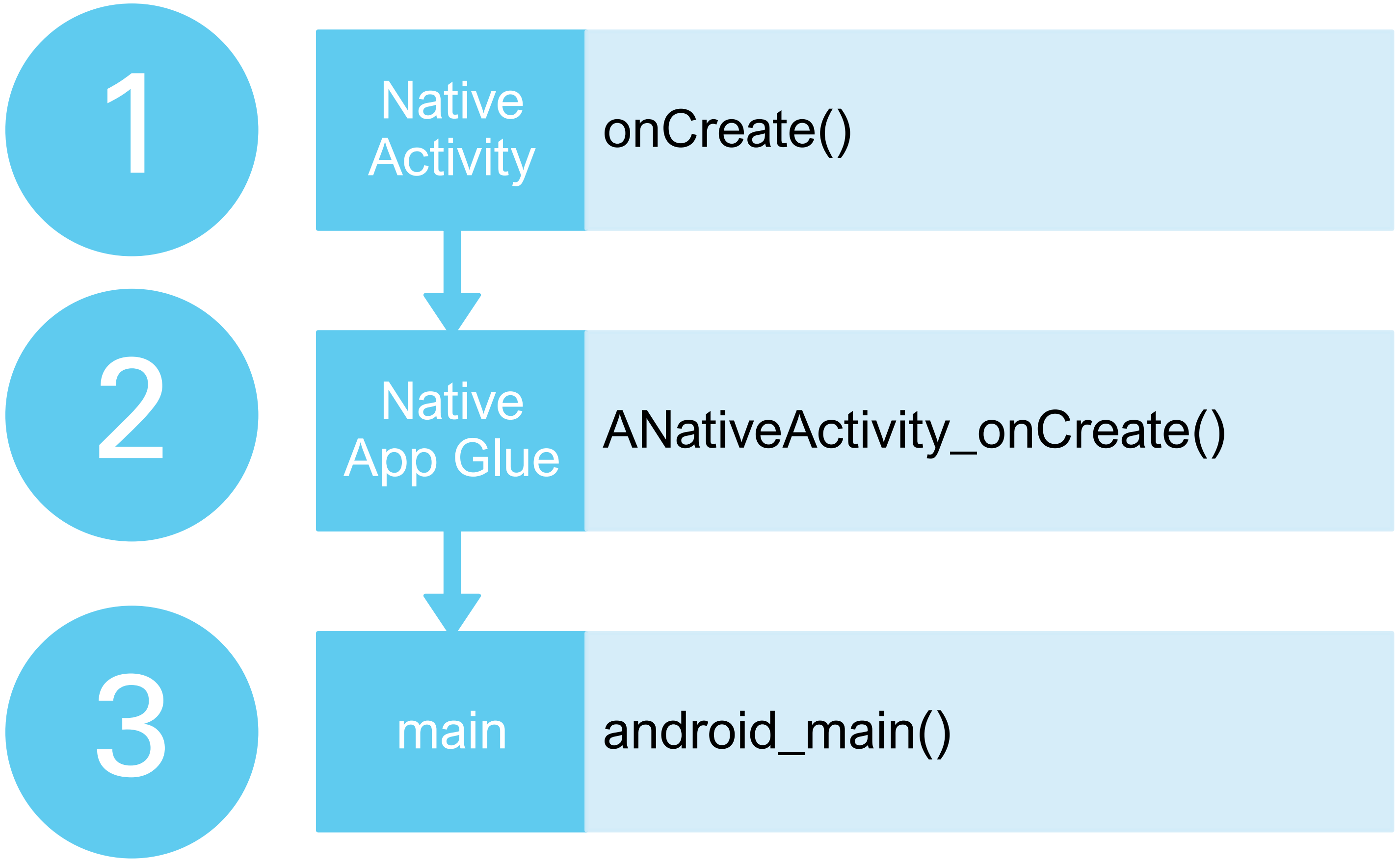
Hands On:
Finding the Standard Entrypoint

Is this possible to bypass?

Android Provides NativeActivity

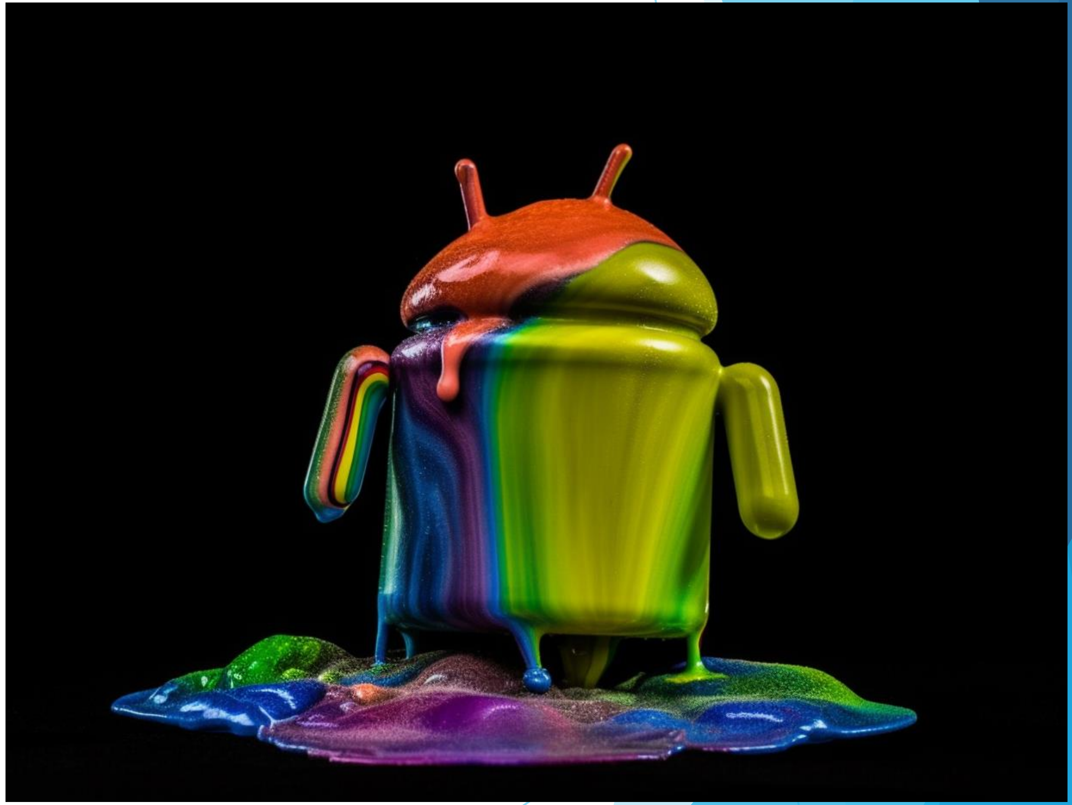
- Helper class provided in Android framework
- Used for gaming apps
- Calls the native library specified in metadata

Android Purely NativeActivity

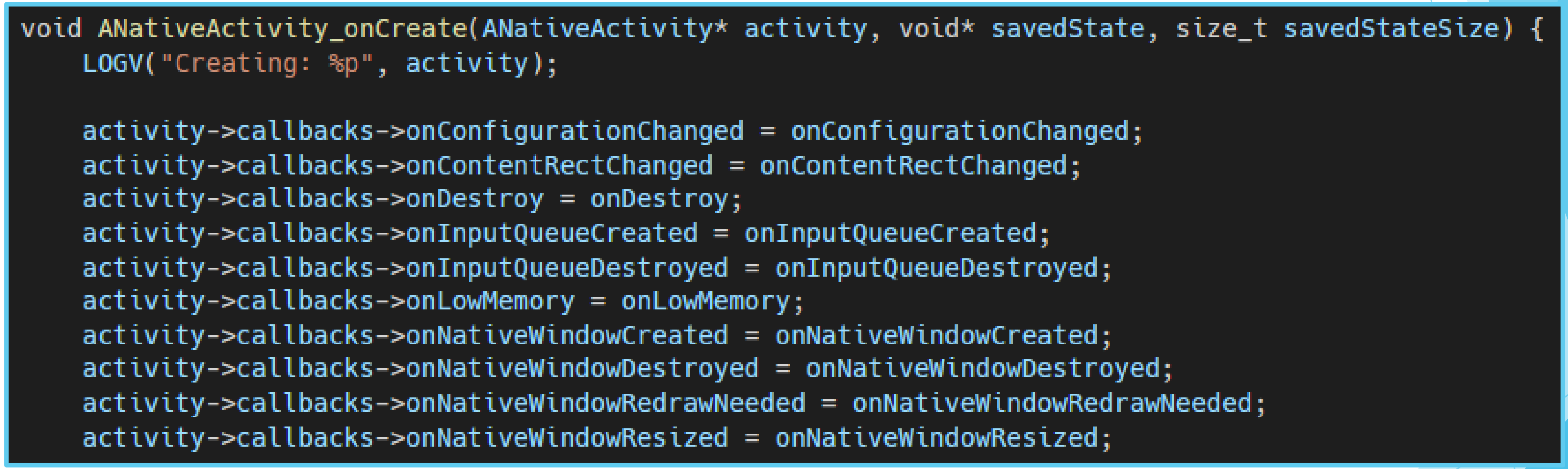


Native Application Glue

- Part of the Android NDK platform code
- Handles application context
- Calls user main
- Defines looper listening for events



Native App Glue Stores Context



```
void ANativeActivity_onCreate(ANativeActivity* activity, void* savedState, size_t savedStateSize) {
    LOGV("Creating: %p", activity);

    activity->callbacks->onConfigurationChanged = onConfigurationChanged;
    activity->callbacks->onContentRectChanged = onContentRectChanged;
    activity->callbacks->onDestroy = onDestroy;
    activity->callbacks->onInputQueueCreated = onInputQueueCreated;
    activity->callbacks->onInputQueueDestroyed = onInputQueueDestroyed;
    activity->callbacks->onLowMemory = onLowMemory;
    activity->callbacks->onNativeWindowCreated = onNativeWindowCreated;
    activity->callbacks->onNativeWindowDestroyed = onNativeWindowDestroyed;
    activity->callbacks->onNativeWindowRedrawNeeded = onNativeWindowRedrawNeeded;
    activity->callbacks->onNativeWindowResized = onNativeWindowResized;
```

User code goes in android_main()

Hands On:
Masking the Entrypoint

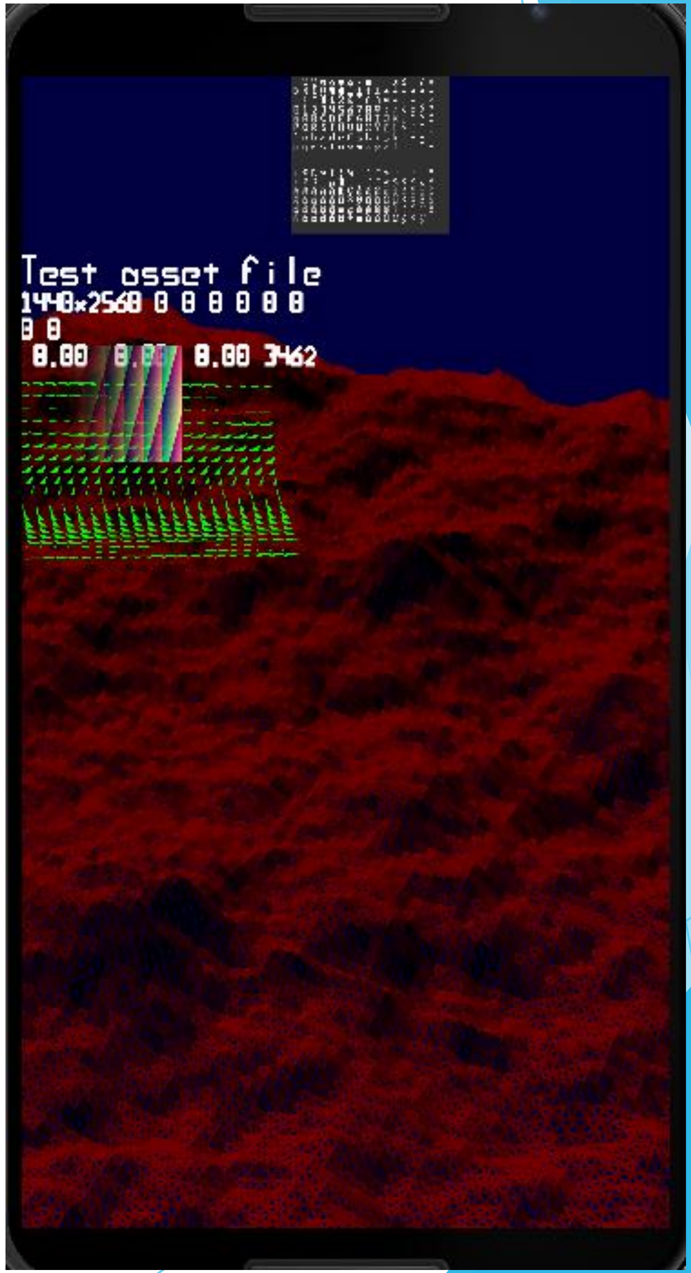
Removing Resource Files

- Optionally remove resources files
  - Android libraries
  - Assets
- Further reduces analysis surfaces

Can we remove the AndroidManifest?

NOPE

Example Purely Native App



```
Test asset file
1440x2560 0 0 0 0 0 0
0 0
 0.00  0.00  0.00 3462
```

App: rawdrawandroid

Excellent, now we can draw malicious shapes!

Translating Java Methods to C++

Standard differences between Java and C++

Java

```java
private void printFibonacci(int N) {
    int num1 = 0;
    int num2 = 1;
    int counter = 0;

    while (counter < N) {
        Log.d("Number", String.valueOf(num1));

        // Calculate next
        int num3 = num2 + num1;
        num1 = num2;
        num2 = num3;
        counter++;
    }
}
```

C++

```cpp
void printFibonacci(int N) {
    int num1 = 0;
    int num2 = 1;
    int counter = 0;

    while (counter < N) {
        __android_log_print(ANDROID_LOG_DEBUG, "Number", "%d", num1);

        // Calculate next
        int num3 = num2 + num1;
        num1 = num2;
        num2 = num3;
        counter++;
    }
}
```

We want to manipulate the device.

Android API Framework

- Library of APIs used by developers
- Callable classes, methods, and variables
- Interface to Android services and hardware

Using the JNI to Invoke Android APIs

- The Android framework is exposed in Java
- JNI is the bridge between Java and C++

Hands On:
Translating Java to Native C++

JNI calls are easy to read / hook.

Further JNI Drawbacks

- Methods are commonly hooked with Frida
- Class names are plaintext strings
- Easy to reverse engineer

What if we want to be stealthier?

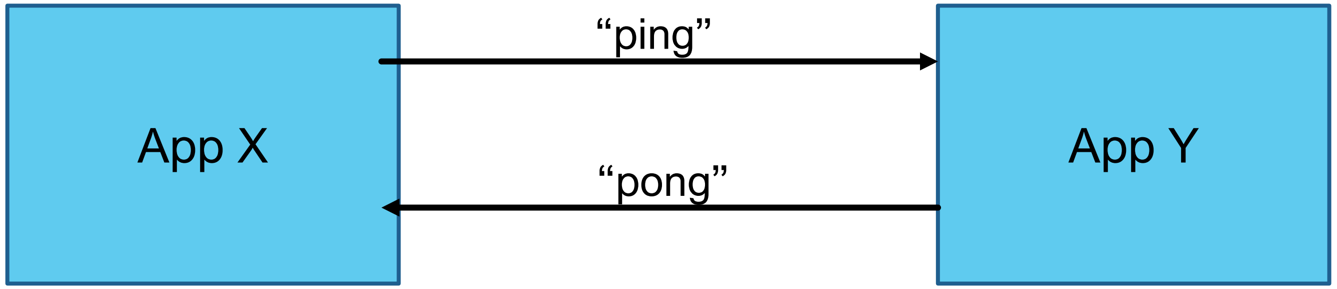
Hiding API Calls Via Binder

Case Study: Dialing a Phone

- Intents send the dial request
- Binder sends this to the TelephonyManager service
- TelephonyManager service handles event

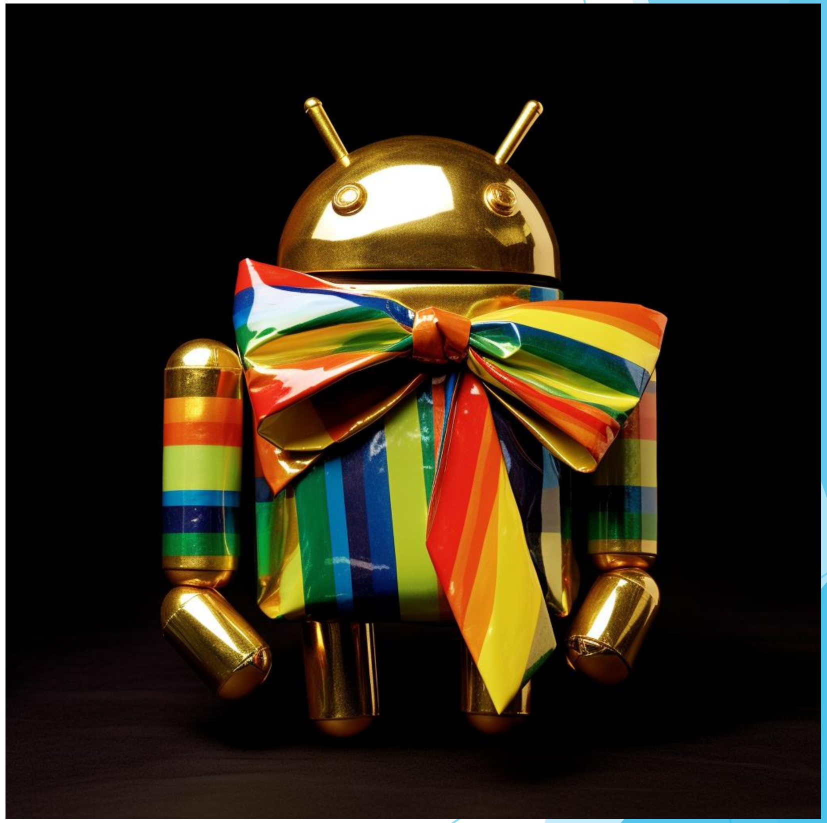
Knowledge time: Exploring the Binder

Enables IPC and RPC in Android



Wrapped by many popular classes

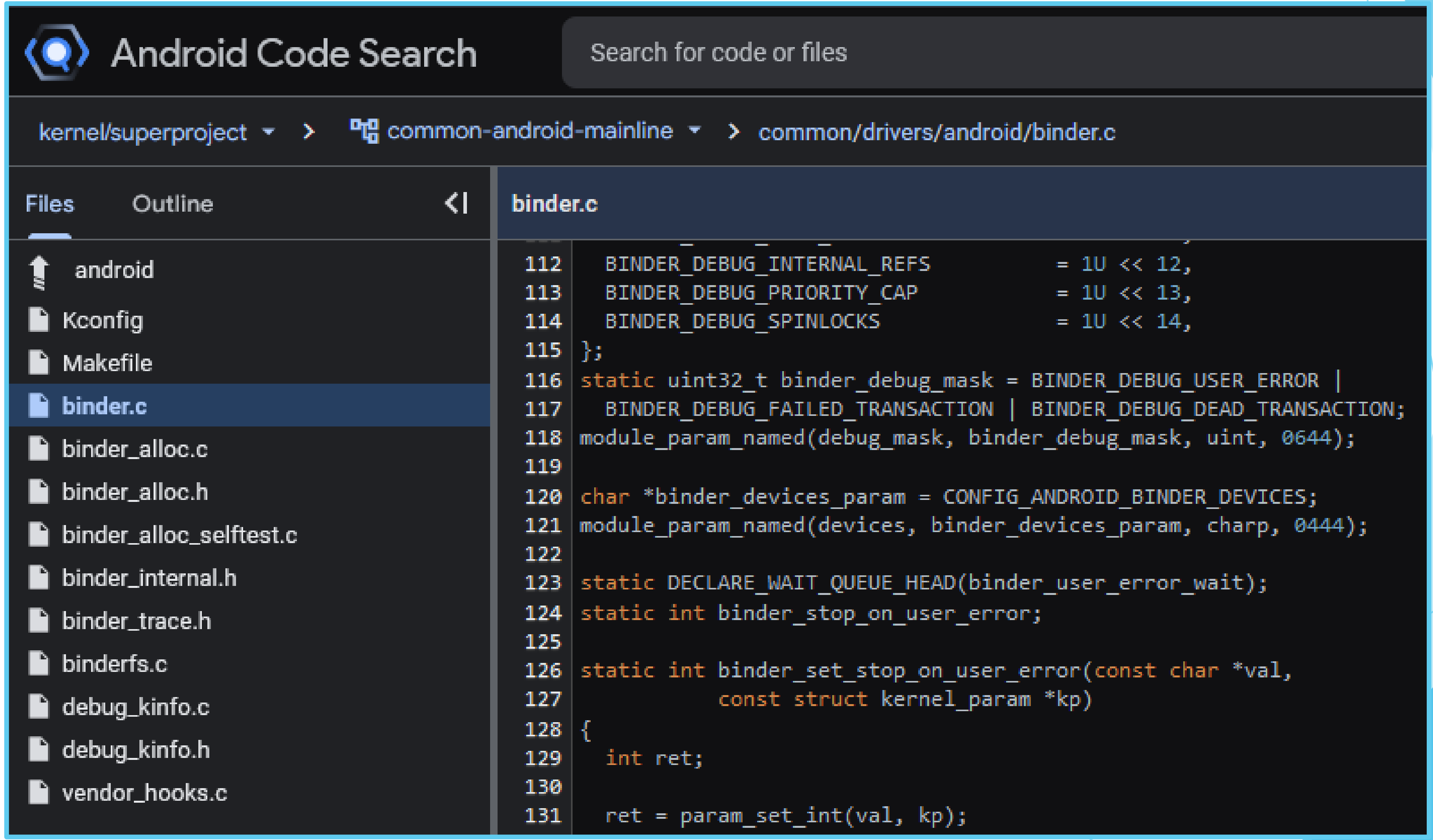
- Intents
- Messengers
- ContentProviders
- Android Interface Definition Language (AIDL)



Can we bypass these common targets?

Let’s dive even deeper

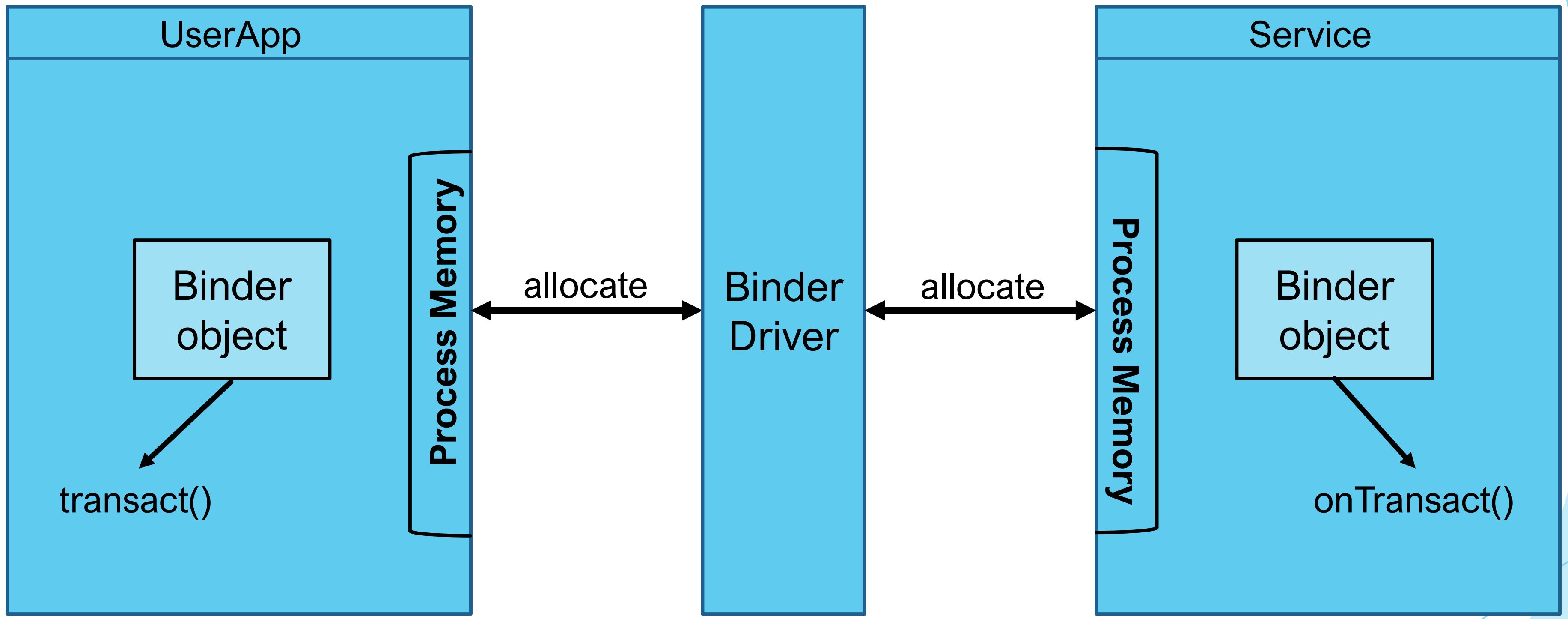
Implemented as a kernel driver



android
Kconfig
Makefile
binder.c
binder_alloc.c
binder_alloc.h
binder_alloc_selftest.c
binder_internal.h
binder_trace.h
binderfs.c
debug_kinfo.c
debug_kinfo.h
vendor_hooks.c

```
  BINDER_DEBUG_INTERNAL_REFS          = 1U << 12,
  BINDER_DEBUG_PRIORITY_CAP           = 1U << 13,
  BINDER_DEBUG_SPINLOCKS              = 1U << 14,
};
static uint32_t binder_debug_mask = BINDER_DEBUG_USER_ERROR |
  BINDER_DEBUG_FAILED_TRANSACTION | BINDER_DEBUG_DEAD_TRANSACTION;
module_param_named(debug_mask, binder_debug_mask, uint, 0644);

char *binder_devices_param = CONFIG_ANDROID_BINDER_DEVICES;
module_param_named(devices, binder_devices_param, charp, 0444);

static DECLARE_WAIT_QUEUE_HEAD(binder_user_error_wait);
static int binder_stop_on_user_error;

static int binder_set_stop_on_user_error(const char *val,
          const struct kernel_param *kp)
{
  int ret;

  ret = param_set_int(val, kp);
```

More Detailed Binder Architecture



Binder Invocation Backend

- Binder allocates memory in target process
- Process handles with onTransact()
- Writes response back
- Binder retrieves and returns response

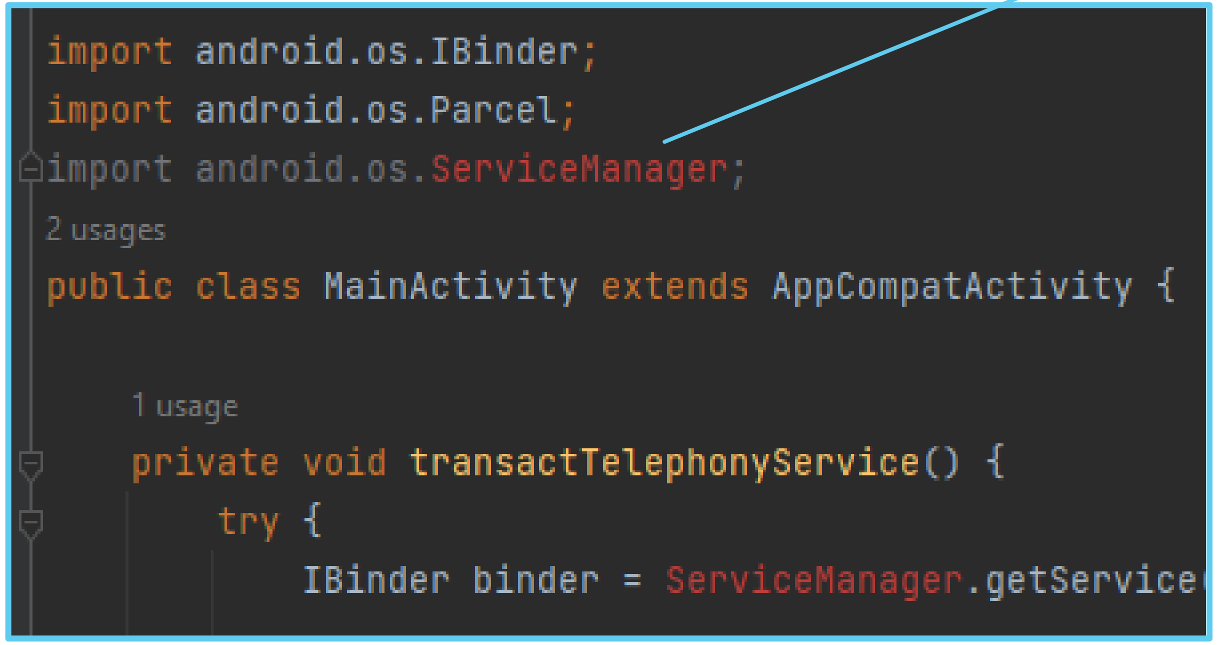
Not required for non-IOCTL commands

We want to transact with existing services.

ServiceManager handles system services.

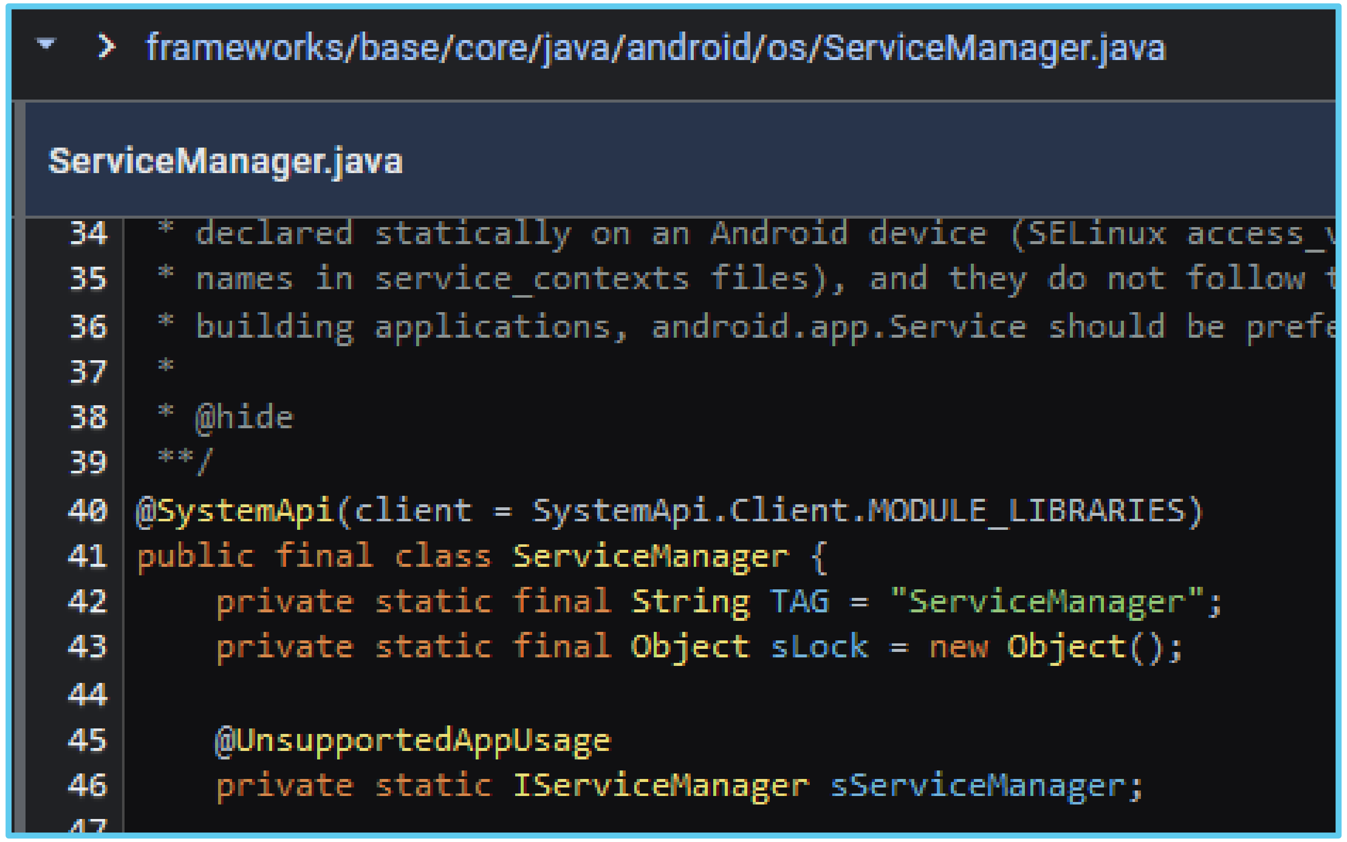
But we can’t use ServiceManager ☹

Cannot resolve symbol



```
import android.os.IBinder;
import android.os.Parcel;
import android.os.ServiceManager;
2 usages
public class MainActivity extends AppCompatActivity {

    1 usage
    private void transactTelephonyService() {
        try {
            IBinder binder = ServiceManager.getService
```

It’s hidden and limited to system use


frameworks/base/core/java/android/os/ServiceManager.java

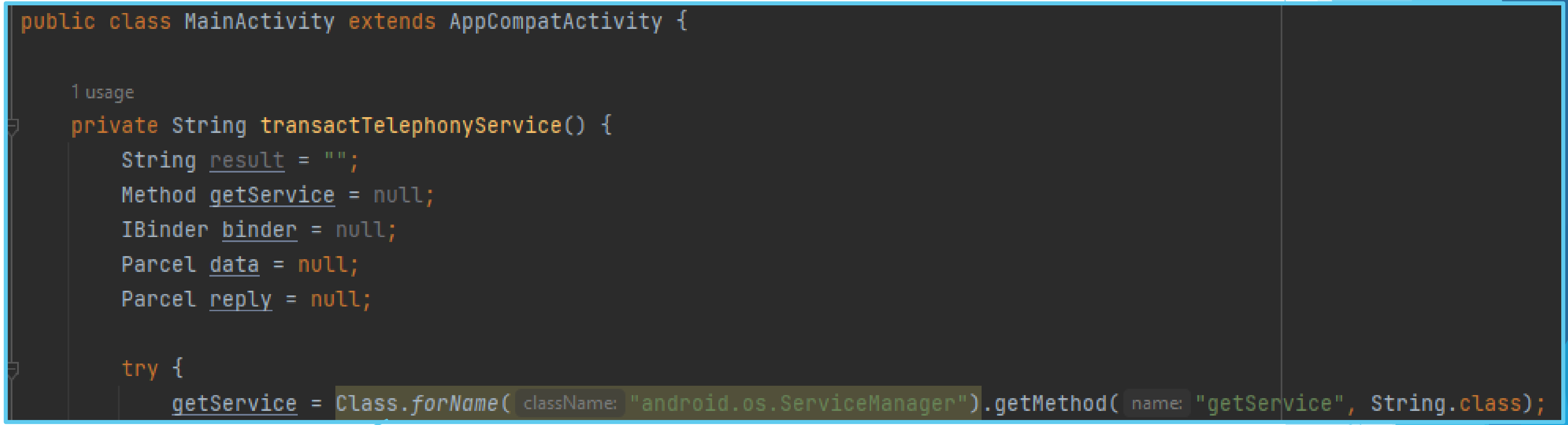
ServiceManager.java

```
 * declared statically on an Android device (SELinux access_v
 * names in service_contexts files), and they do not follow t
 * building applications, android.app.Service should be prefe
 *
 * @hide
 **/
@SystemApi(client = SystemApi.Client.MODULE_LIBRARIES)
public final class ServiceManager {
    private static final String TAG = "ServiceManager";
    private static final Object sLock = new Object();

    @UnsupportedAppUsage
    private static IServiceManager sServiceManager;
```

Or is it?

Reflection doesn’t respect hidden APIs lol

No more errors!



```
public class MainActivity extends AppCompatActivity {

    1 usage
    private String transactTelephonyService() {
        String result = "";
        Method getService = null;
        IBinder binder = null;
        Parcel data = null;
        Parcel reply = null;

        try {
            getService = Class.forName( className: "android.os.ServiceManager").getMethod( name: "getService", String.class);
```

Reflection

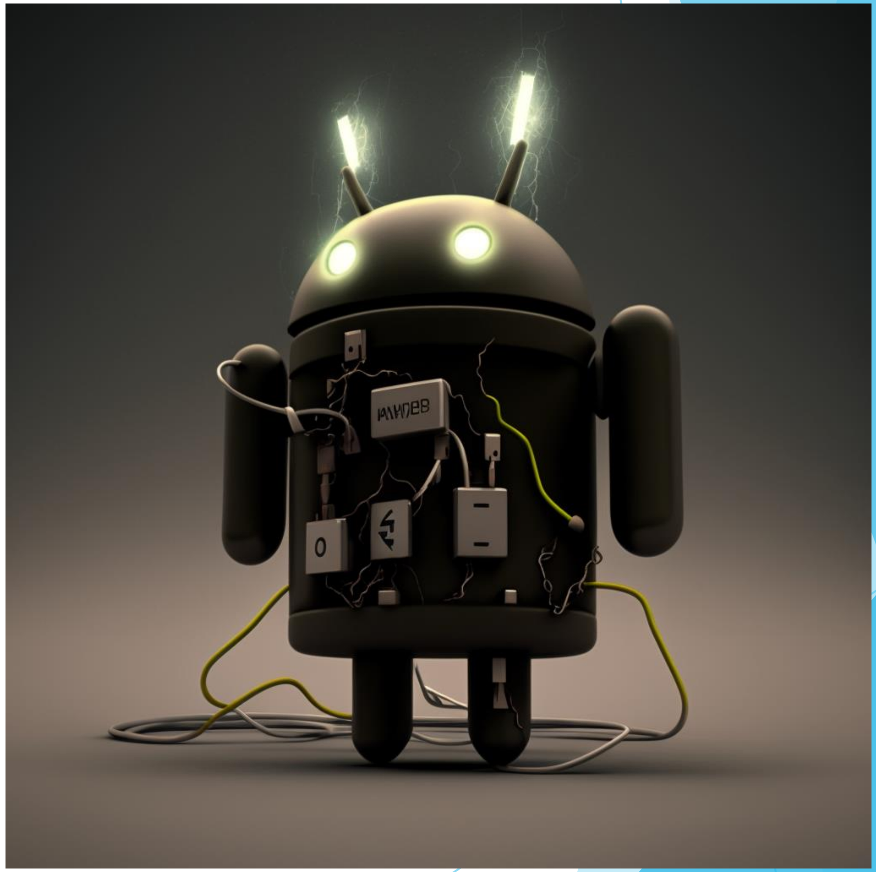
Hands On:
Finding Callable Services

Bound Invokable services

- ActivityManagerService
- LocationManager
- PackageManagerService
- TelephonyManager
- SensorService
- WifiManagerNotificationManager
- AudioManager
- PowerManagerWindowManager
- ClipboardServiceInputMethodManager
- AlarmManagerBatteryManager
- StorageManagerConnectivityManager
- BluetoothManagerVibratorService
- UserManager
- AccessibilityManager

Use Parcel objects to transmit data

- Container for messages
- Requires target interface
- Must write method arguments



Call transact() to communicate

- Transmits the message
- Receives the response via another Parcel
- Requires method code instead of name

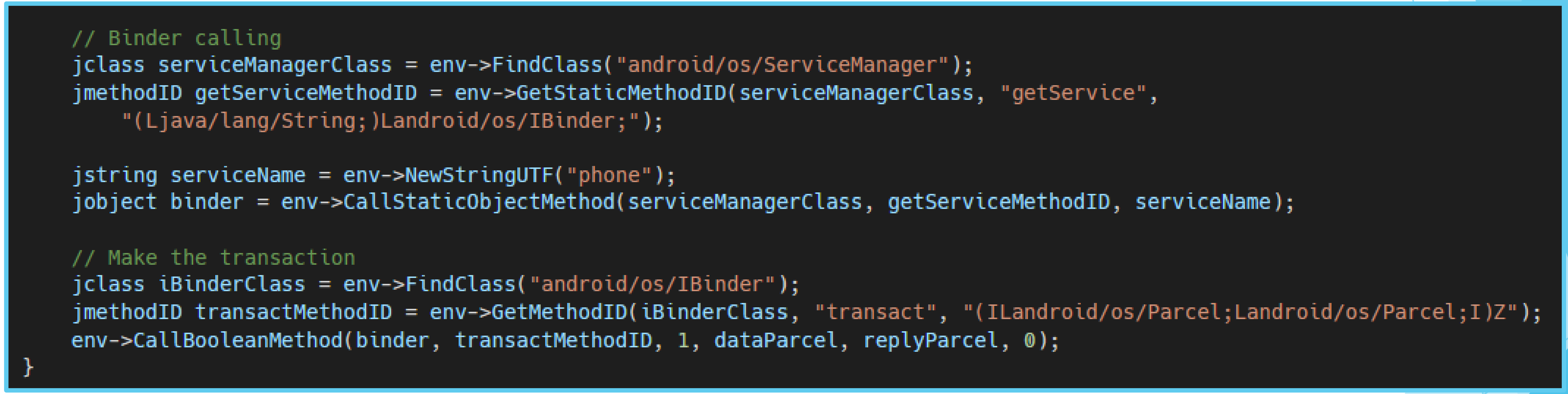
Hands On:
Invoking dial via Binder

We need to do this in C++.

I won't make you translate it all.

Phew!

We already know native translation



```
    // Binder calling
    jclass serviceManagerClass = env->FindClass("android/os/ServiceManager");
    jmethodID getServiceMethodID = env->GetStaticMethodID(serviceManagerClass, "getService",
        "(Ljava/lang/String;)Landroid/os/IBinder;");

    jstring serviceName = env->NewStringUTF("phone");
    jobject binder = env->CallStaticObjectMethod(serviceManagerClass, getServiceMethodID, serviceName);

    // Make the transaction
    jclass iBinderClass = env->FindClass("android/os/IBinder");
    jmethodID transactMethodID = env->GetMethodID(iBinderClass, "transact", "(ILandroid/os/Parcel;Landroid/os/Parcel;I)Z");
    env->CallBooleanMethod(binder, transactMethodID, 1, dataParcel, replyParcel, 0);
}
```

Test time!

Encrypt class strings for further protection

- Encrypt string targets of reflective calls
- Avoid plaintext
  - Target services
  - Target class names

Summarizing Our Obfuscation

Purely Native Code

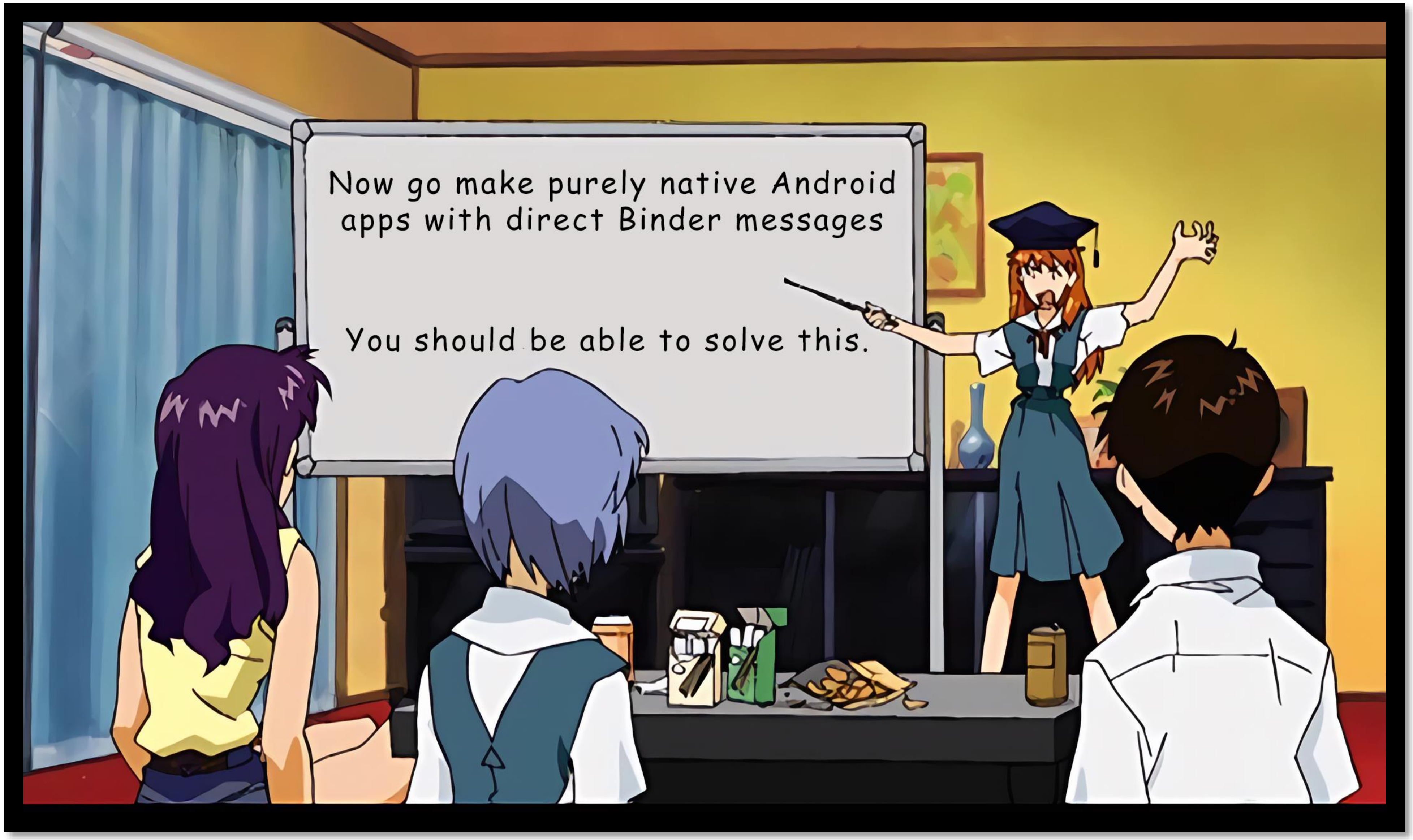
- Masks the entrypoint
- No pretty Java code
- Challenges automated and human analysts

Direct Binder Invocation

- Use for system service calls
- Entirely avoids method names
- Bypasses hooks

Will we see this more?

It’s hard to write lol


Now go make purely native Android apps with direct Binder messages
You should be able to solve this.

Thank you!

Bonus Section

References

LaurieWired TROOPERS23
Github Repo

https://github.com/LaurieWired/AndroidPurelyNative_Troopers23

Android Native Code Resources

- Sample: native-activity
  - https://developer.android.com/ndk/samples/sample_na
- Android framework NativeActivity class
  - https://android.googlesource.com/platform/frameworks/base.git/+/master/core/java/android/app/NativeActivity.java
- Rawdrawandroid
  - https://github.com/cnlohr/rawdrawandroid

Java Native Interface (JNI)

- JNI Functions
- JNI Types and Signatures

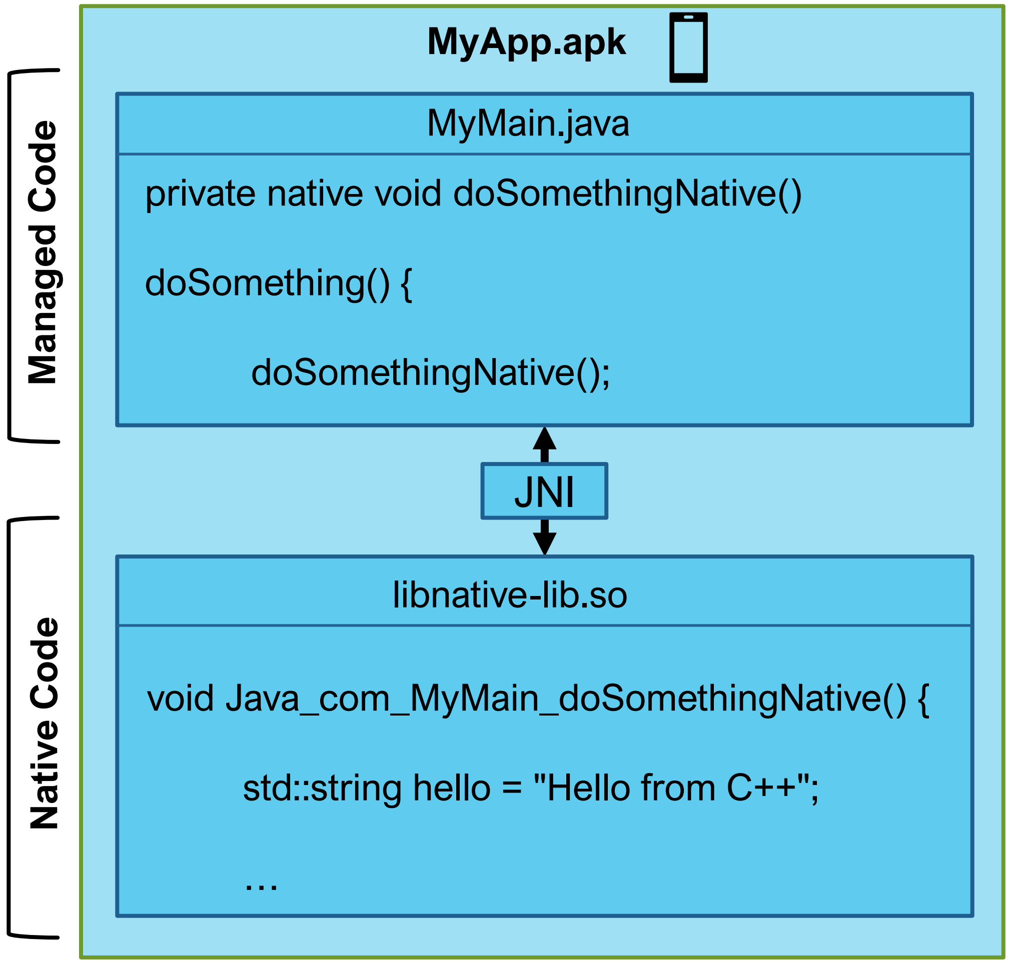
Binder

- Documentation
  - https://developer.android.com/reference/android/os/Binder
- Source code
  - https://cs.android.com/android/kernel/superproject/+/common-android-mainline:common/drivers/android/binder.c

ServiceManager

- Source code
  - https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/java/android/os/ServiceManager.java

Native App Structure



```
private native void doSomethingNative()

doSomething() {

        doSomethingNative();
```

```
void Java_com_MyMain_doSomethingNative() {

        std::string hello = "Hello from C++";

        …
```

We can use NativeActivity as main

Entrypoint

User code library

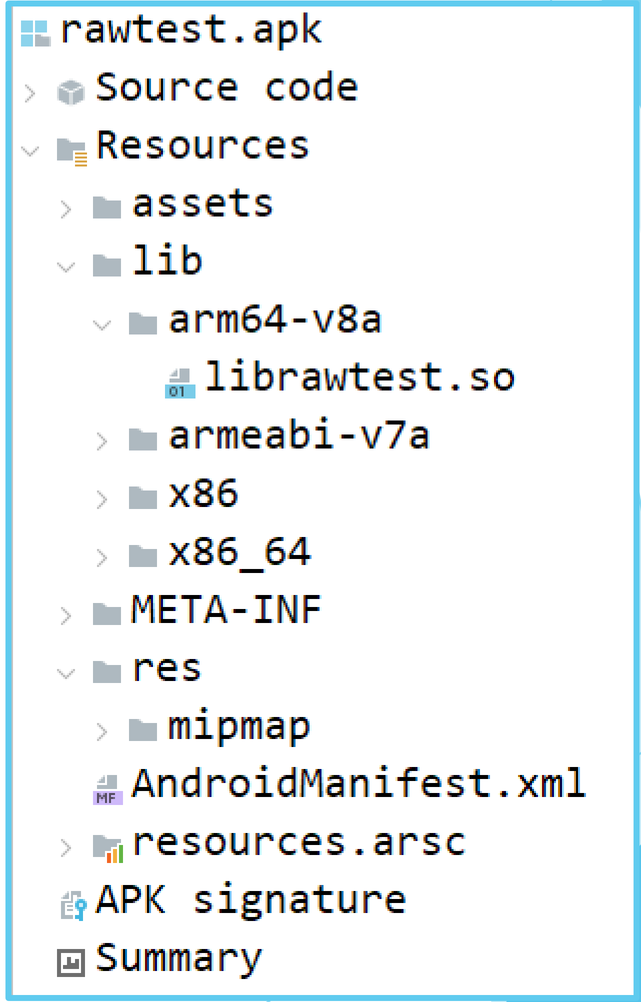
```
<activity android:name="android.app.NativeActivity" android:label="@string/app_name"
    android:configChanges="orientation|keyboardHidden">

    <meta-data android:name="android.app.lib_name" android:value="native-activity" />

    <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Define Obfuscation

- Obfuscation obscures app data and functionality
- Essential for Android
- Decompiled into pretty Java code

Native Code in Android

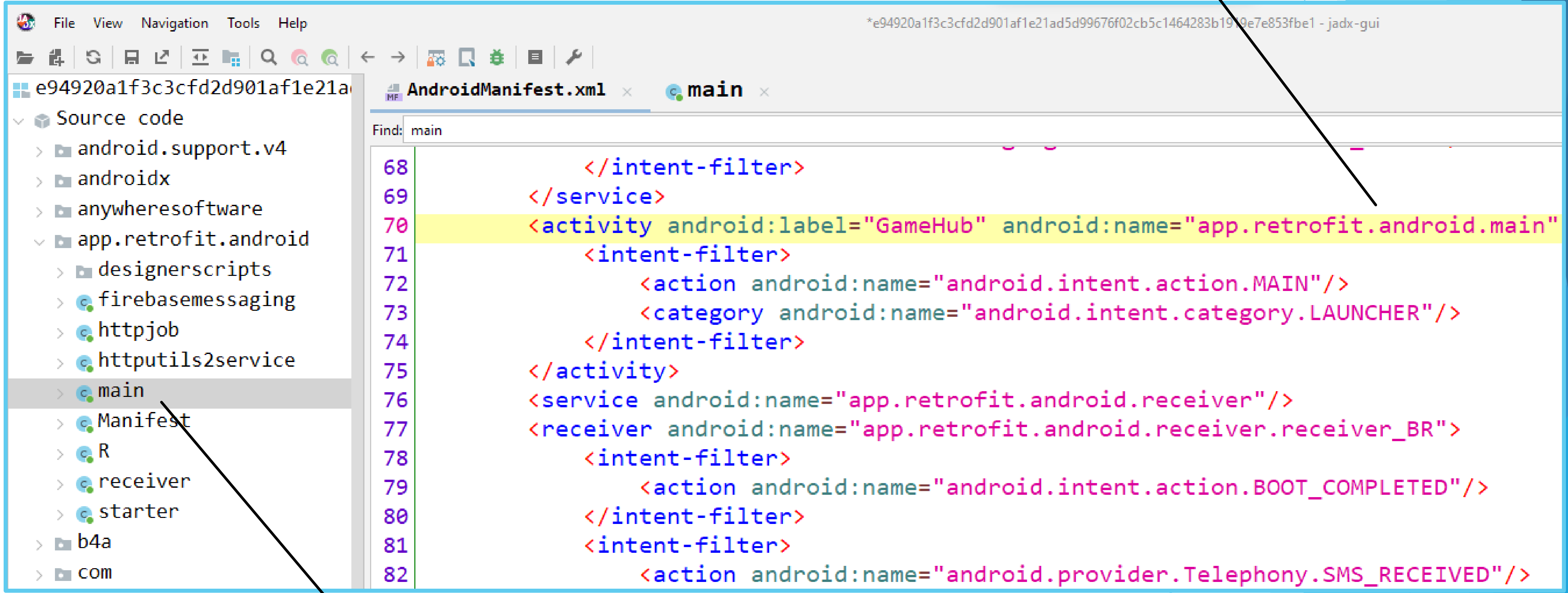
- Implemented as Linux ELF binaries
- Shared object (.so) files
- Compiled to run on particular instruction set architectures



```
rawtest.apk
> Source code
v Resources
  > assets
  v lib
    v arm64-v8a
        librawtest.so
    > armeabi-v7a
    > x86
    > x86_64
  > META-INF
  v res
    > mipmap
    AndroidManifest.xml
  > resources.arsc
  APK signature
  Summary
```

Standard Entrypoint Recognition

Java entrypoint



```
            </intent-filter>
        </service>
        <activity android:label="GameHub" android:name="app.retrofit.android.main"
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service android:name="app.retrofit.android.receiver"/>
        <receiver android:name="app.retrofit.android.receiver.receiver_BR">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
```

Java implementation

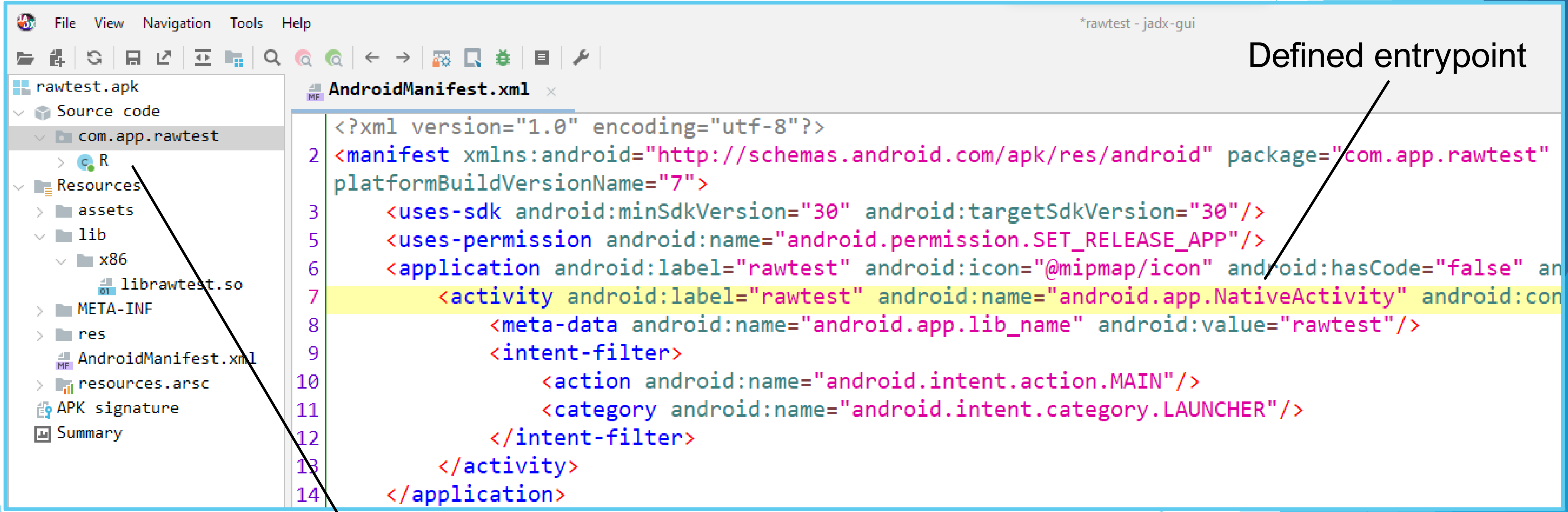
Android NDK

- NDK stands for Native Development Kit
- Contains tools for writing C/C++ code in Android

NativeActivity

- Runs in the main app thread
- Managed code entrypoint
- Sets up and loads user native library

Minimal Native APK

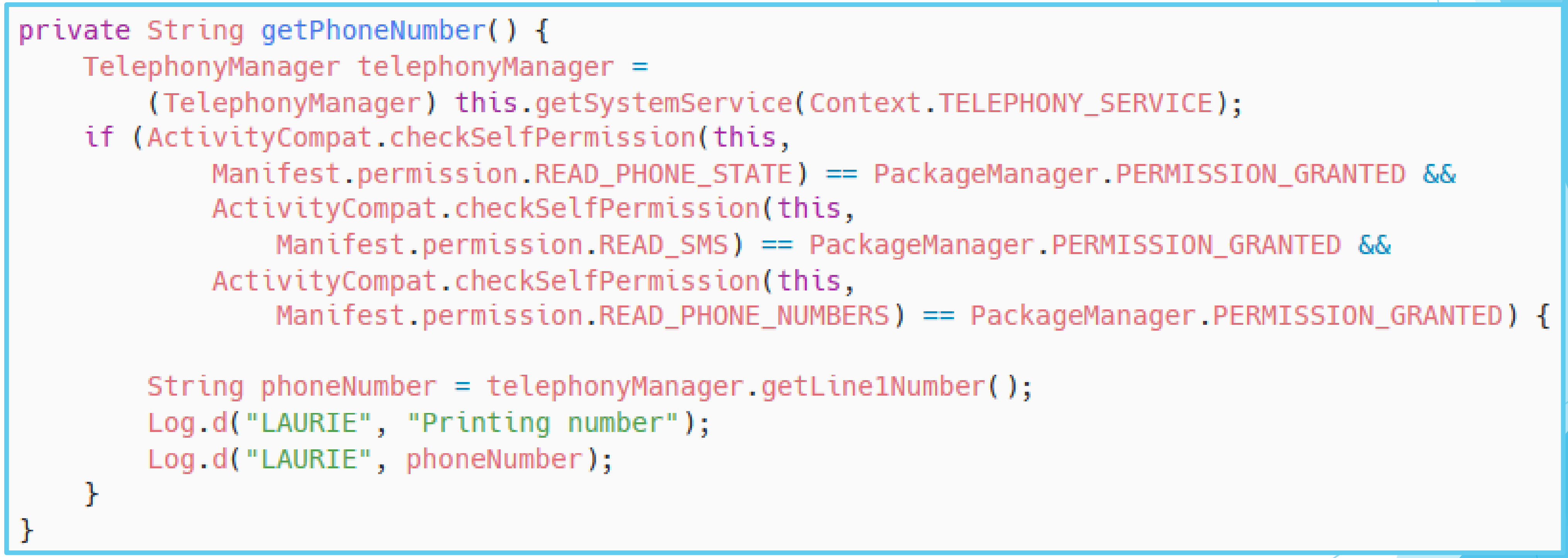


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.app.rawtest"
platformBuildVersionName="7">
    <uses-sdk android:minSdkVersion="30" android:targetSdkVersion="30"/>
    <uses-permission android:name="android.permission.SET_RELEASE_APP"/>
    <application android:label="rawtest" android:icon="@mipmap/icon" android:hasCode="false" an
        <activity android:label="rawtest" android:name="android.app.NativeActivity" android:con
            <meta-data android:name="android.app.lib_name" android:value="rawtest"/>
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
```

Defined entrypoint

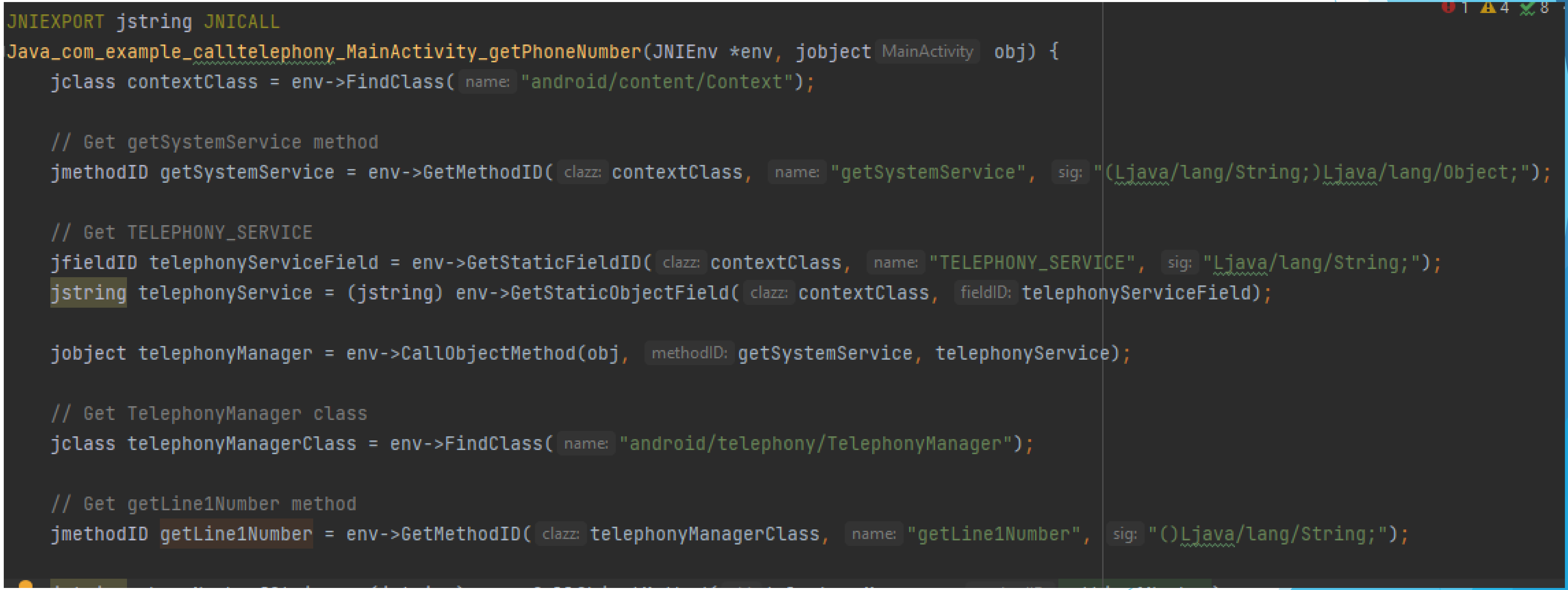
No such class!

Java Method



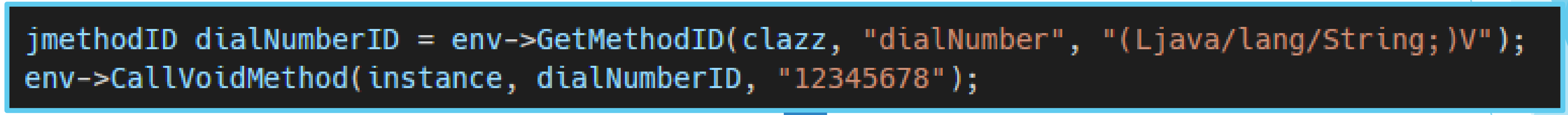
```java
private String getPhoneNumber() {
    TelephonyManager telephonyManager =
        (TelephonyManager) this.getSystemService(Context.TELEPHONY_SERVICE);
    if (ActivityCompat.checkSelfPermission(this,
            Manifest.permission.READ_PHONE_STATE) == PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
                Manifest.permission.READ_SMS) == PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
                Manifest.permission.READ_PHONE_NUMBERS) == PackageManager.PERMISSION_GRANTED) {

        String phoneNumber = telephonyManager.getLine1Number();
        Log.d("LAURIE", "Printing number");
        Log.d("LAURIE", phoneNumber);
    }
}
```

Equivalent C++ Method (fix picture)

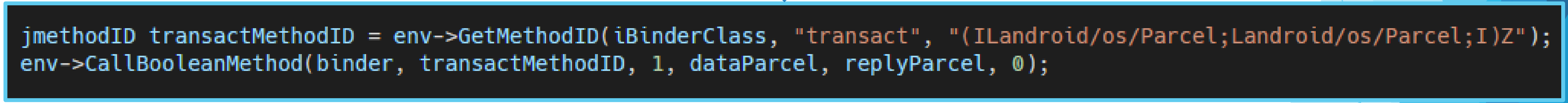


```
JNIEXPORT jstring JNICALL
Java_com_example_calltelephony_MainActivity_getPhoneNumber(JNIEnv *env, jobject MainActivity obj) {
    jclass contextClass = env->FindClass( name: "android/content/Context");

    // Get getSystemService method
    jmethodID getSystemService = env->GetMethodID( clazz: contextClass,  name: "getSystemService",  sig: "(Ljava/lang/String;)Ljava/lang/Object;");

    // Get TELEPHONY_SERVICE
    jfieldID telephonyServiceField = env->GetStaticFieldID( clazz: contextClass,  name: "TELEPHONY_SERVICE",  sig: "Ljava/lang/String;");
    jstring telephonyService = (jstring) env->GetStaticObjectField( clazz: contextClass,  fieldID: telephonyServiceField);

    jobject telephonyManager = env->CallObjectMethod(obj,  methodID: getSystemService, telephonyService);

    // Get TelephonyManager class
    jclass telephonyManagerClass = env->FindClass( name: "android/telephony/TelephonyManager");

    // Get getLine1Number method
    jmethodID getLine1Number = env->GetMethodID( clazz: telephonyManagerClass,  name: "getLine1Number",  sig: "()Ljava/lang/String;");
```

Let’s make the first line look more like the second.



```
jmethodID dialNumberID = env->GetMethodID(clazz, "dialNumber", "(Ljava/lang/String;)V");
env->CallVoidMethod(instance, dialNumberID, "12345678");
```



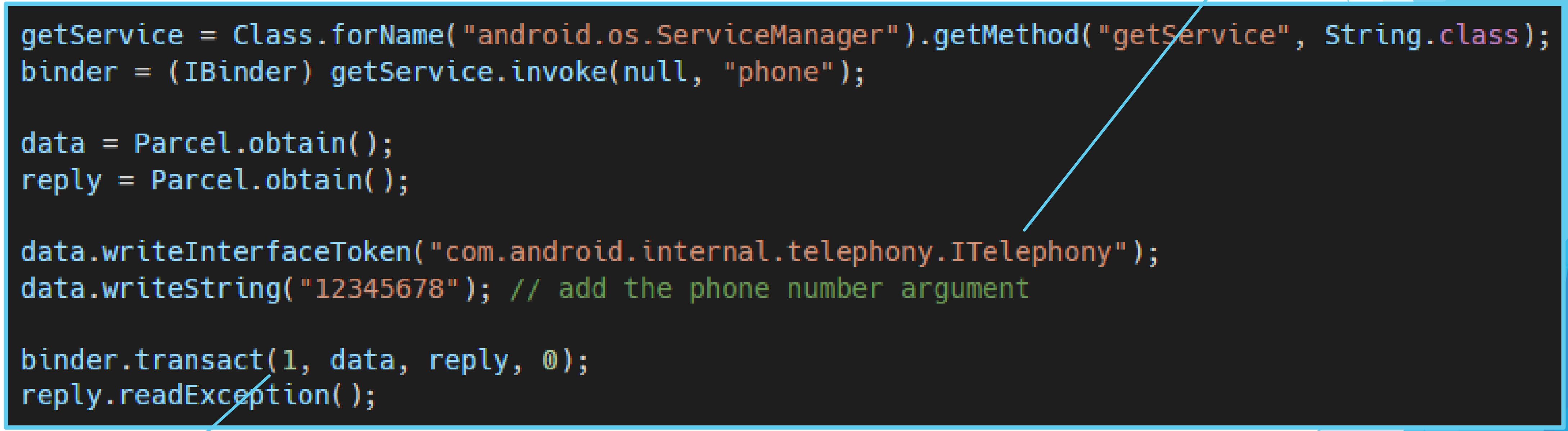
```
jmethodID transactMethodID = env->GetMethodID(iBinderClass, "transact", "(ILandroid/os/Parcel;Landroid/os/Parcel;I)Z");
env->CallBooleanMethod(binder, transactMethodID, 1, dataParcel, replyParcel, 0);
```

Binding to System Services with ServiceManager

- Returns Binder object for target service
- Manages system services
- Limited to system usage

Simple Java Transaction Example

System interface



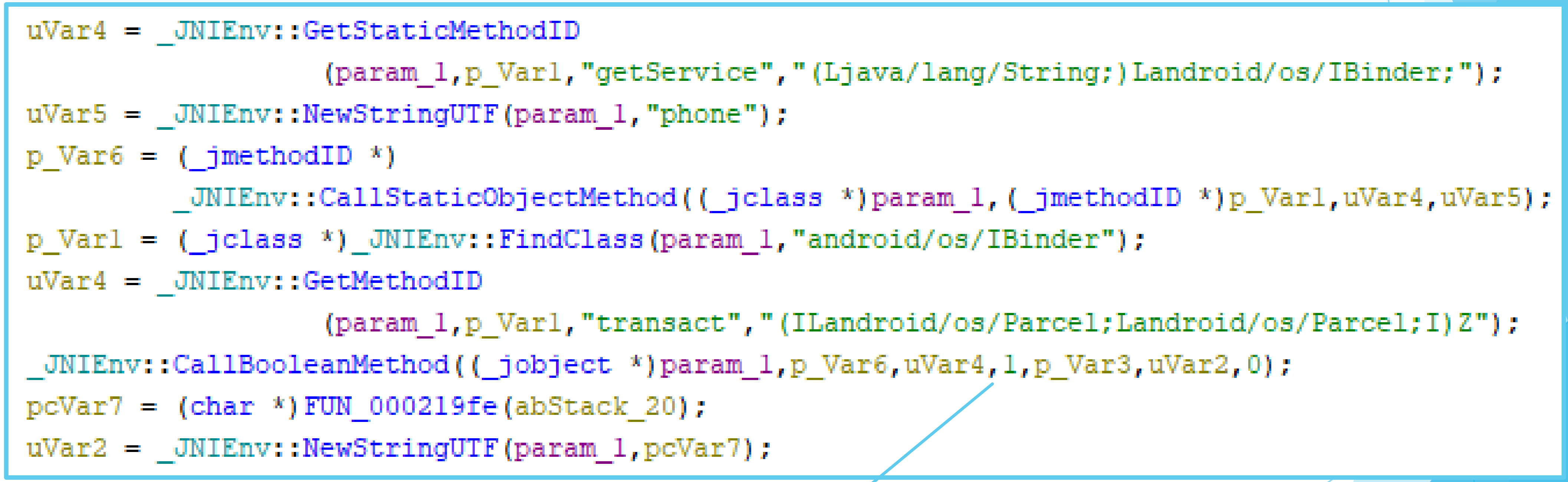
```
getService = Class.forName("android.os.ServiceManager").getMethod("getService", String.class);
binder = (IBinder) getService.invoke(null, "phone");

data = Parcel.obtain();
reply = Parcel.obtain();

data.writeInterfaceToken("com.android.internal.telephony.ITelephony");
data.writeString("12345678"); // add the phone number argument

binder.transact(1, data, reply, 0);
reply.readException();
```

Method code

Service Constant Mappings

- TELEPHONY_SERVICE = "phone";
- TELECOM_SERVICE = "telecom";
- CARRIER_CONFIG_SERVICE = "carrier_config";
- EUICC_SERVICE = "euicc";
- EUICC_CARD_SERVICE = "euicc_card";
- MMS_SERVICE = "mms";
- CLIPBOARD_SERVICE = "clipboard";
- TEXT_CLASSIFICATION_SERVICE = "textclassification";
- SELECTION_TOOLBAR_SERVICE = "selection_toolbar";
- FONT_SERVICE = "font";
- ATTENTION_SERVICE = "attention";
- ROTATION_RESOLVER_SERVICE = "resolver";

Resulting Code in Ghidra



```
uVar4 = _JNIEnv::GetStaticMethodID
                  (param_1,p_Var1,"getService","(Ljava/lang/String;)Landroid/os/IBinder;");
uVar5 = _JNIEnv::NewStringUTF(param_1,"phone");
p_Var6 = (_jmethodID *)
         _JNIEnv::CallStaticObjectMethod((_jclass *)param_1,(_jmethodID *)p_Var1,uVar4,uVar5);
p_Var1 = (_jclass *)_JNIEnv::FindClass(param_1,"android/os/IBinder");
uVar4 = _JNIEnv::GetMethodID
                  (param_1,p_Var1,"transact","(ILandroid/os/Parcel;Landroid/os/Parcel;I)Z");
_JNIEnv::CallBooleanMethod((_jobject *)param_1,p_Var6,uVar4,1,p_Var3,uVar2,0);
pcVar7 = (char *)FUN_000219fe(abStack_20);
uVar2 = _JNIEnv::NewStringUTF(param_1,pcVar7);
```

???