



# Secure Android Application

## Defend Customer Data

(In development)

[baolq@hpt.vn](mailto:baolq@hpt.vn)

@BaoLQOnSecurity

VIETNAM SECURITY DAY 2015

- Web/Mobile Pentester at HPT
- CEH, eMAPT certified
- Guitar, Coffee, Traveling, Panda and ...

## Agenda

- Imagine a World without Mobile?
- Android Overview
- Defend Customer Data
- Q & A



Imagine a World without Mobile?



Knowing IT

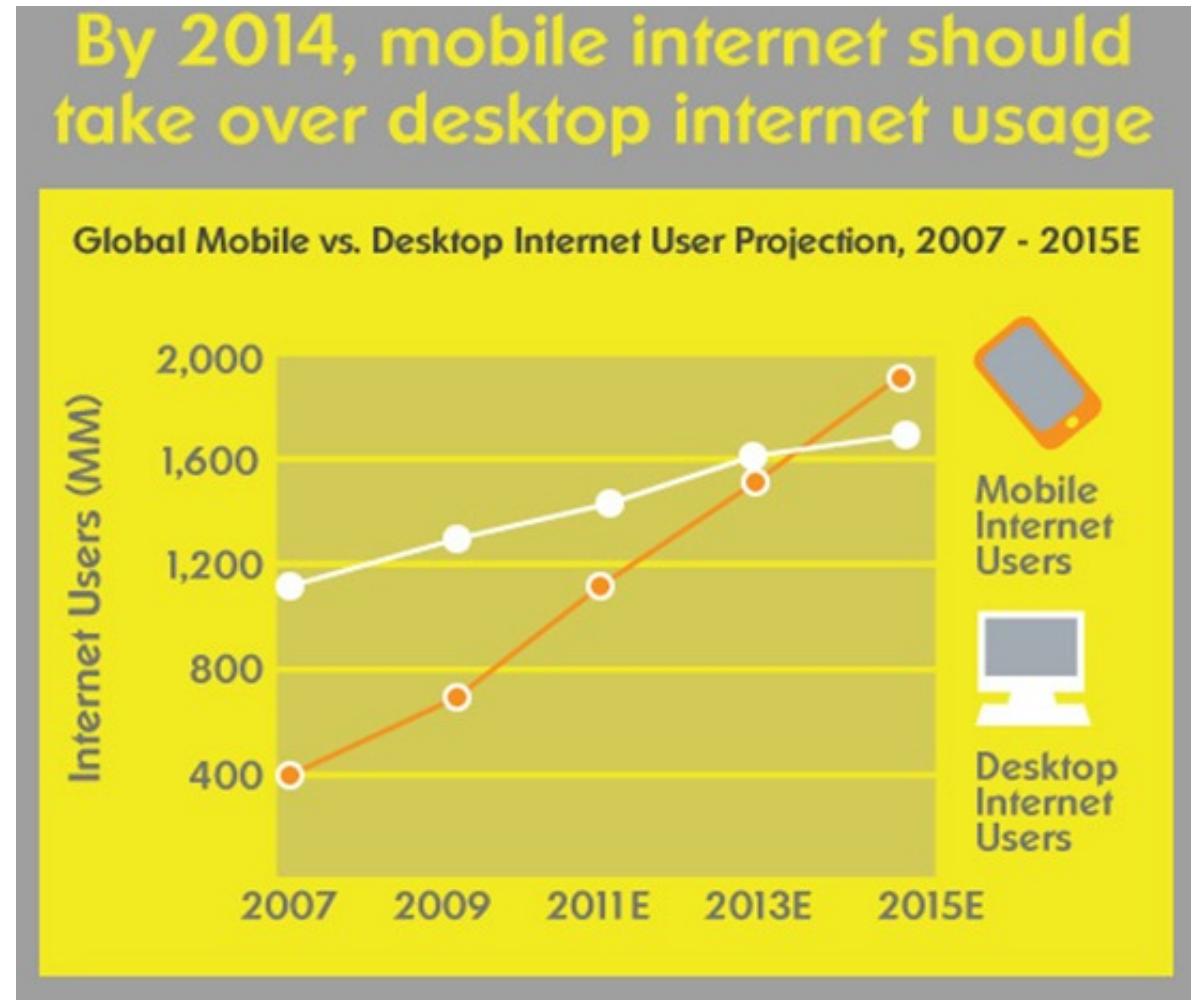
## “Awesome” Figures

It's my super hero! It changed my life!

- The age of Ultron (Mobile Devices)!



<http://www.smartinsights.com/>

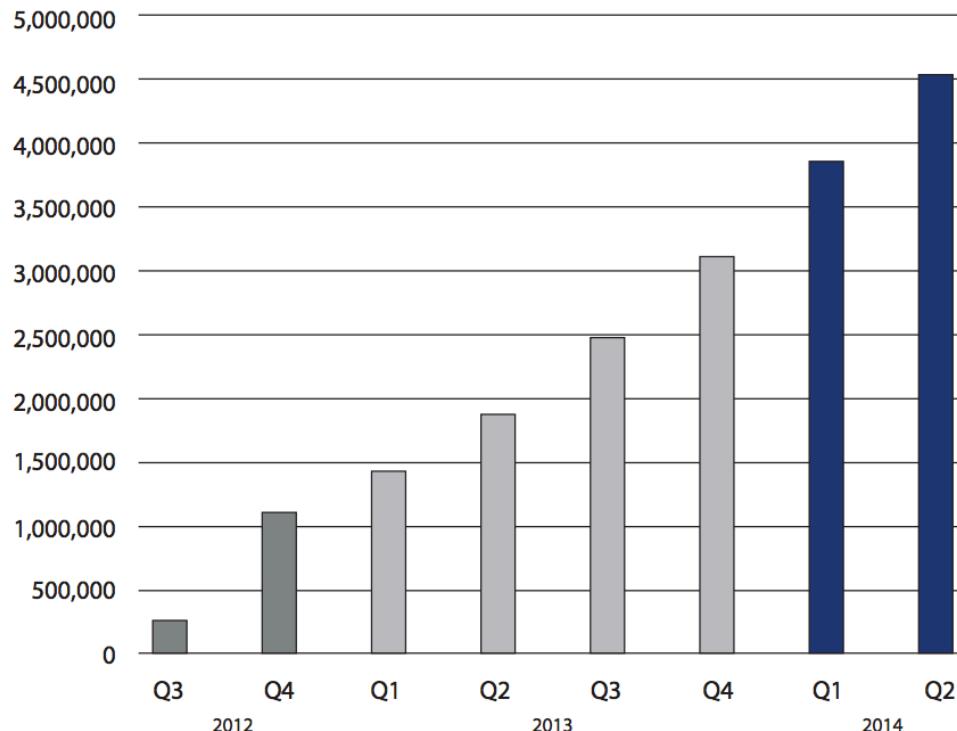


## McAfee Mobile Malware Report

Total count of McAfee's mobile malware increased by 17% in Q2 2014

Significant number of these malware exploits were designed to exploit digital wallet service & popular messaging app. Mobile malware samples grew by 167 percent between Q1 2013 and Q1 2014.

Total count of McAfee's mobile malware increased by 17% in Q2 2014.

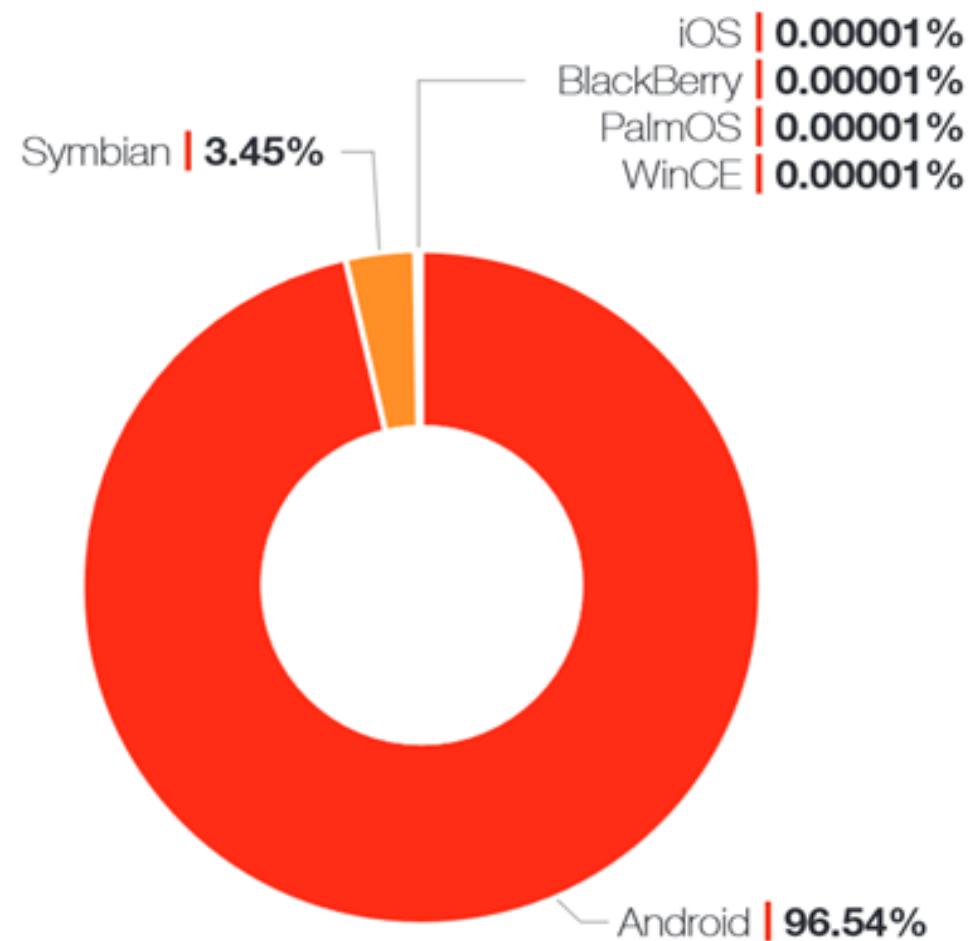


**167%**

**2013 - 2014**

Think about a Device you bring everyday?

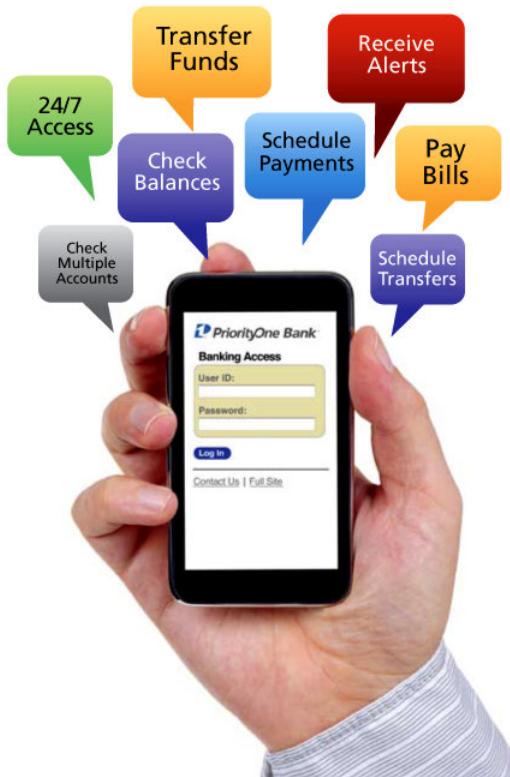
- The more popular – The most targeted



*Mobile Malware - Fortinet results (2014 report)*

## So scary, leave my Money alone?

- How secure your Bank's app dev can ensure?

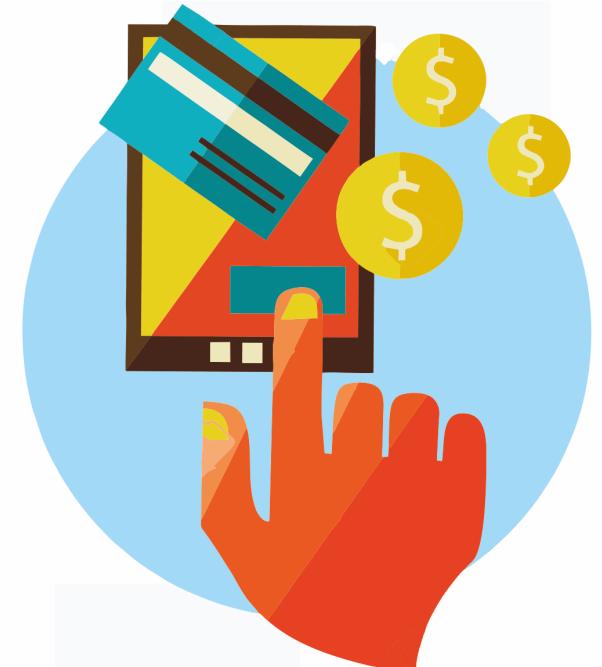


Security Report of  
Top 100 Mobile Banking Apps



# 70%

Mobile Banking Apps of  
Top 100 Banks in India & APAC  
are Vulnerable to Security  
Attacks & Data Leaks

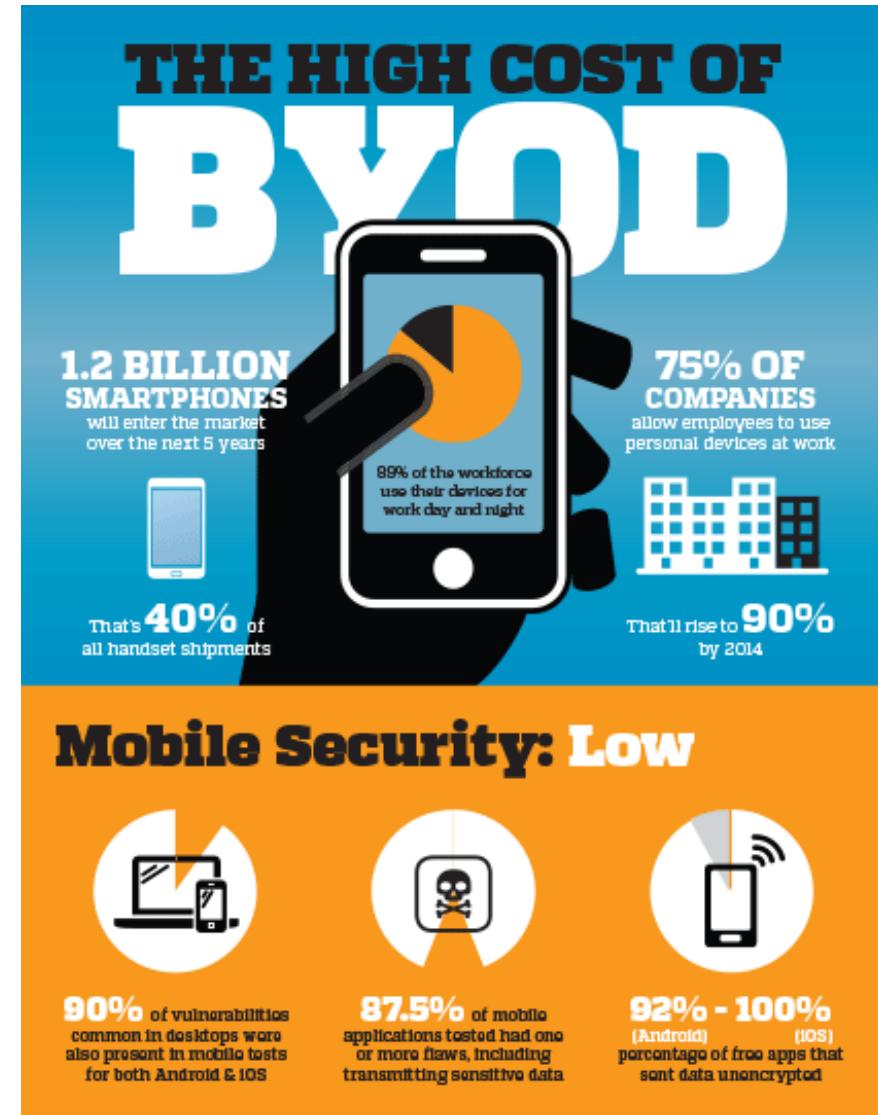


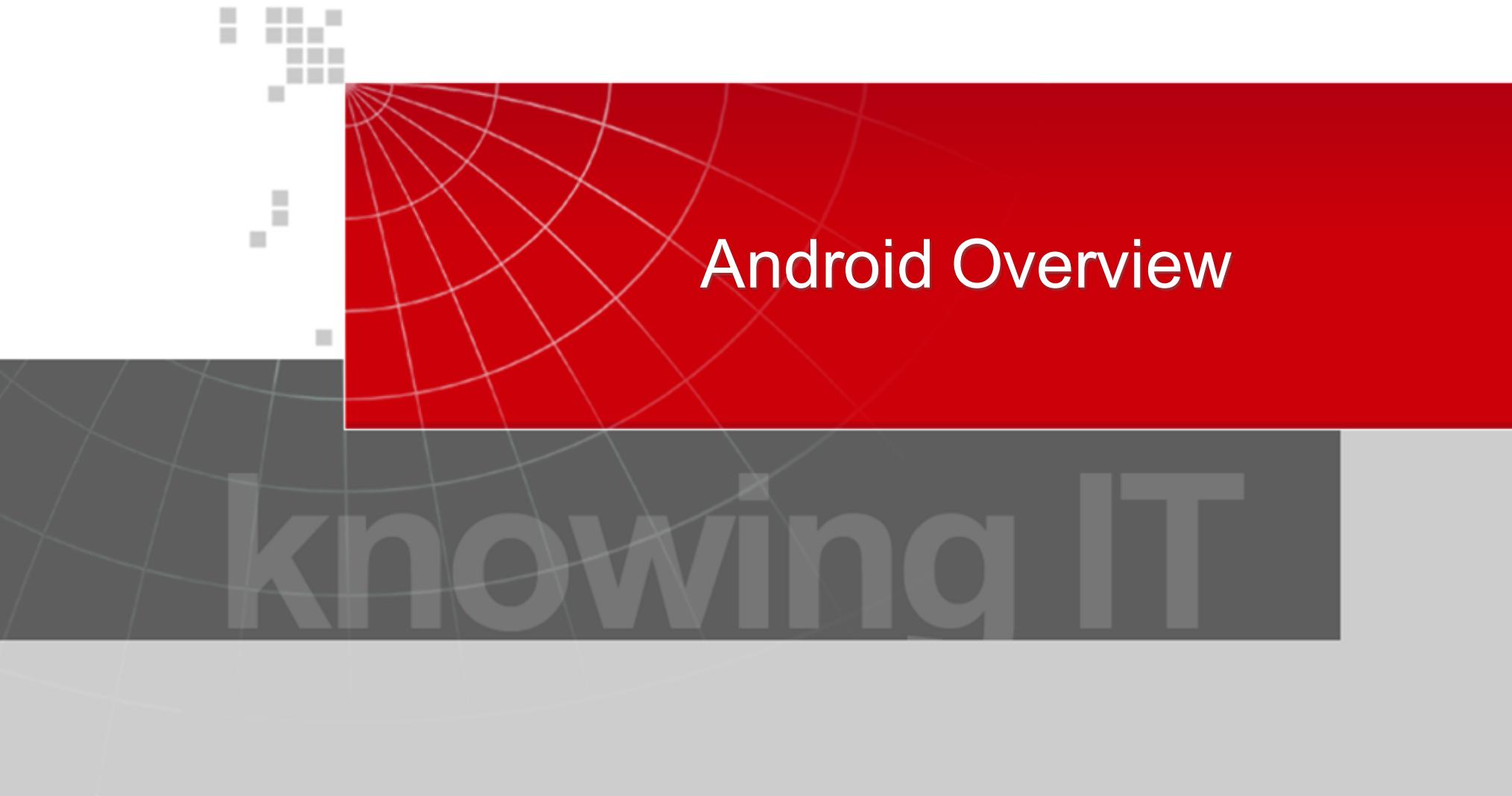
## What's the effective Enterprise solution?

- Employee Devices



- Security





A decorative graphic in the background features a red rectangular area with a white grid pattern. To its left is a gray rectangular area with a white grid pattern. A vertical column of small gray squares is positioned between these two sections. The overall design has a technical, network-like aesthetic.

## Android Overview

**knowing IT**

- Android OS & Architecture Model
- App Directory
- App Components
- Access Controls

They are too delicious! aren't they?



**Cupcake**  
Android 1.5



**Donut**  
Android 1.6



**Eclair**  
Android 2.0/2.1



**Froyo**  
Android 2.2.x



**Gingerbread**  
Android 2.3.x



**Honeycomb**  
Android 3.x



**Ice Cream Sandwich**  
Android 4.0.x



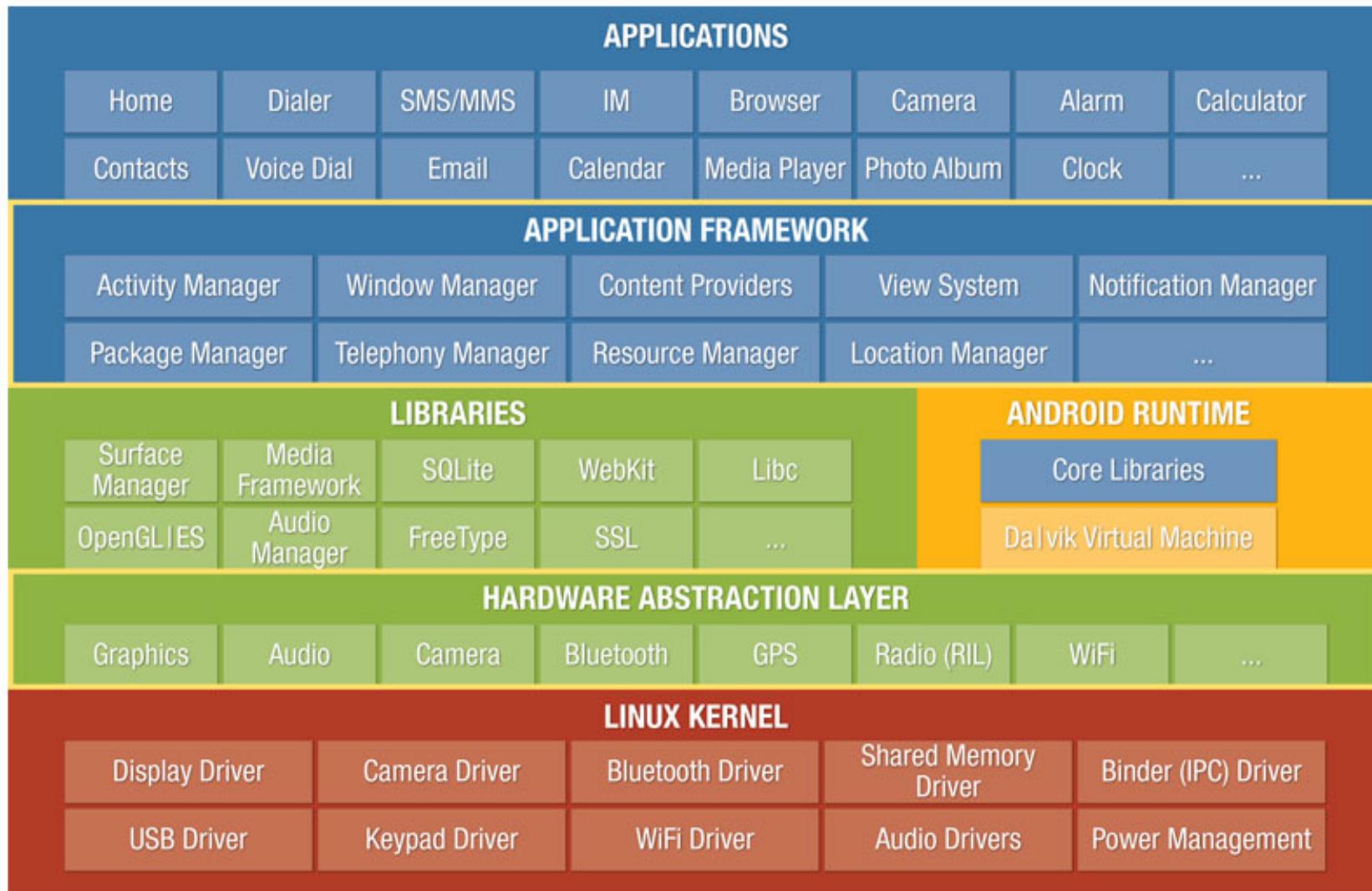
**Jelly Bean**  
Android 4.1.x



**KitKat**  
Android 4.4.x



**Lollipop**  
Android 5.0

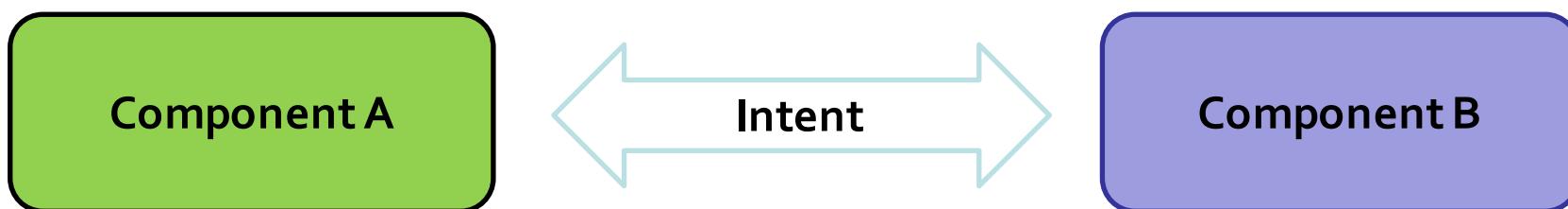


/data/data/<package\_name>/

- **AndroidManifest.xml**: declare components, permissions, version, package, ...
- Lib folder: native library .so, .dll (C/C++, C#)
- **classes.dex**: like exe
- Res folder: contains resource not compile
- Assets folder: retrieved by AssessManager

We use Intent to communicate each other

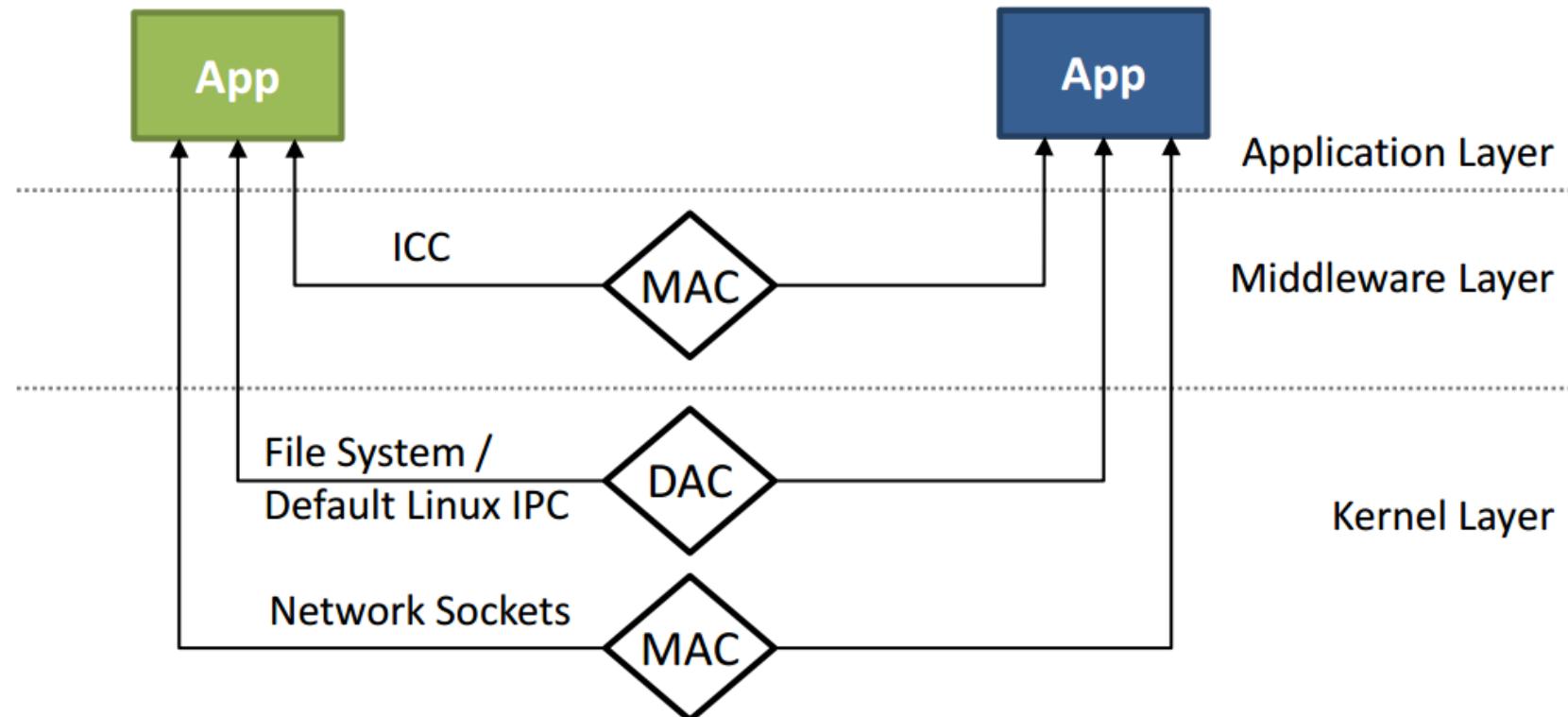
- **Activity:** GUI
- **Service:** Background process
- **Content Provider:** SQLite database
- **Broadcast Receiver:** responds to system-wide broadcast announcements



## Linux-based OS

- **Discretionary Access Control (DAC)** – used to isolate apps (Linux)
- **Mandatory Access Control (MAC)** – used to mediate the establishment of inter-component/application communication

## Diff between 2 concepts



ICC – Inter-Component Communication (or IPC)  
DAC – Discretionary Access Control  
MAC – Mandatory Access Control



A decorative graphic element on the left side of the slide. It consists of a dark gray rectangular area containing a light gray grid pattern. Above this, a red rectangular area contains a white grid pattern. To the left of the red area, there is a vertical column of small, light gray squares of varying sizes, creating a pixelated effect.

## Security Features



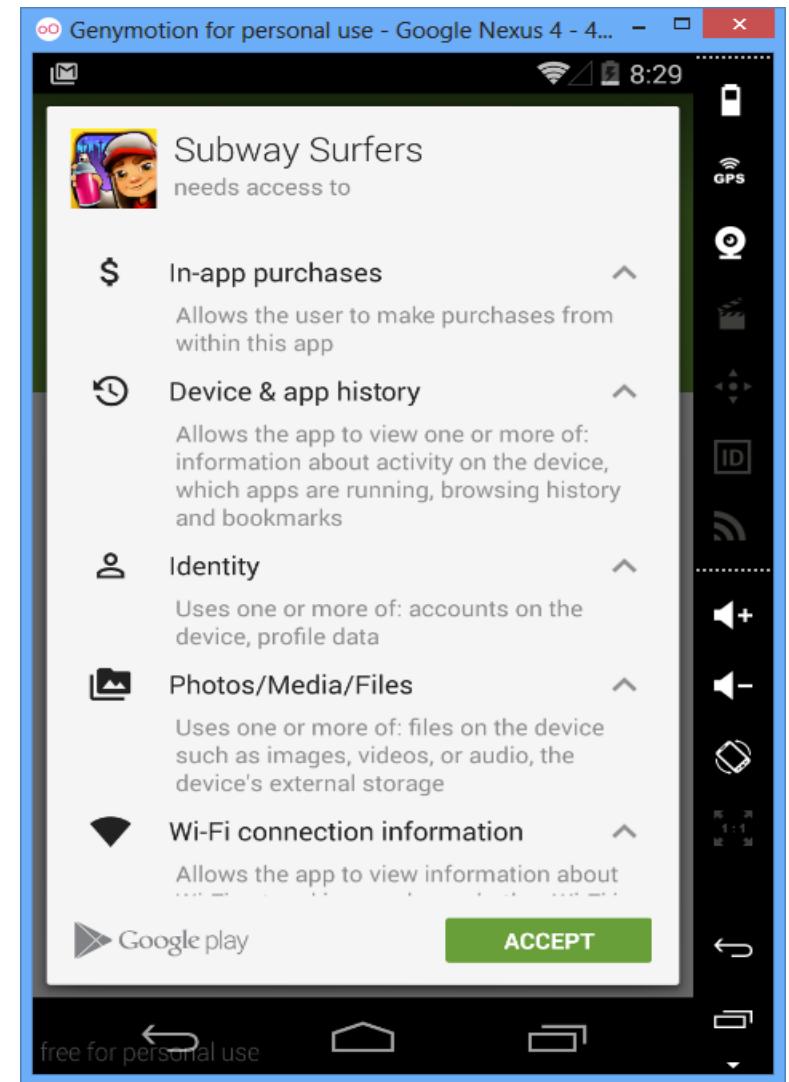
A large, semi-transparent gray rectangular area containing the words "knowing IT" in a large, bold, gray sans-serif font. This text is partially obscured by the red grid graphic above it.

- Permissions
- App Signing
- App Verification
- App Sandbox

## User-granted Permission

Protected APIs (OS layer):

- Camera functions
- Location data (GPS)
- Bluetooth functions
- Telephony functions
- SMS/MMS functions
- Network/data connections



# Application-defined Permission

```
<permission android:name="com.hpt.MY_PERMISSION"  
    android:description="@string/des_permission"  
    android:label="mycustom"  
</permission> // define permission  
<uses-permission android:name="com.hpt.MY_PERMISSION"/> // for  
external app usage  
<activity android:permission="com.hpt.MY_PERMISSION"  
    android:name=".YourActivity"  
    android:label="@string/activity_label" /> // apply to activity
```

## Protection Level

Normal	Dangerous	Signature	SignatureOrSystem
<ul style="list-style-type: none"><li>• System automatically grants at installation</li><li>• Don't warn to user</li></ul>	<ul style="list-style-type: none"><li>• Need user confirmation when install</li><li>• Don't need to match signature</li></ul>	<ul style="list-style-type: none"><li>• System automatically grants the permission</li><li>• Should be signed by the same Certificate</li></ul>	<ul style="list-style-type: none"><li>• Same certificate</li><li>• Grants only to Applications in the Android system image</li></ul>

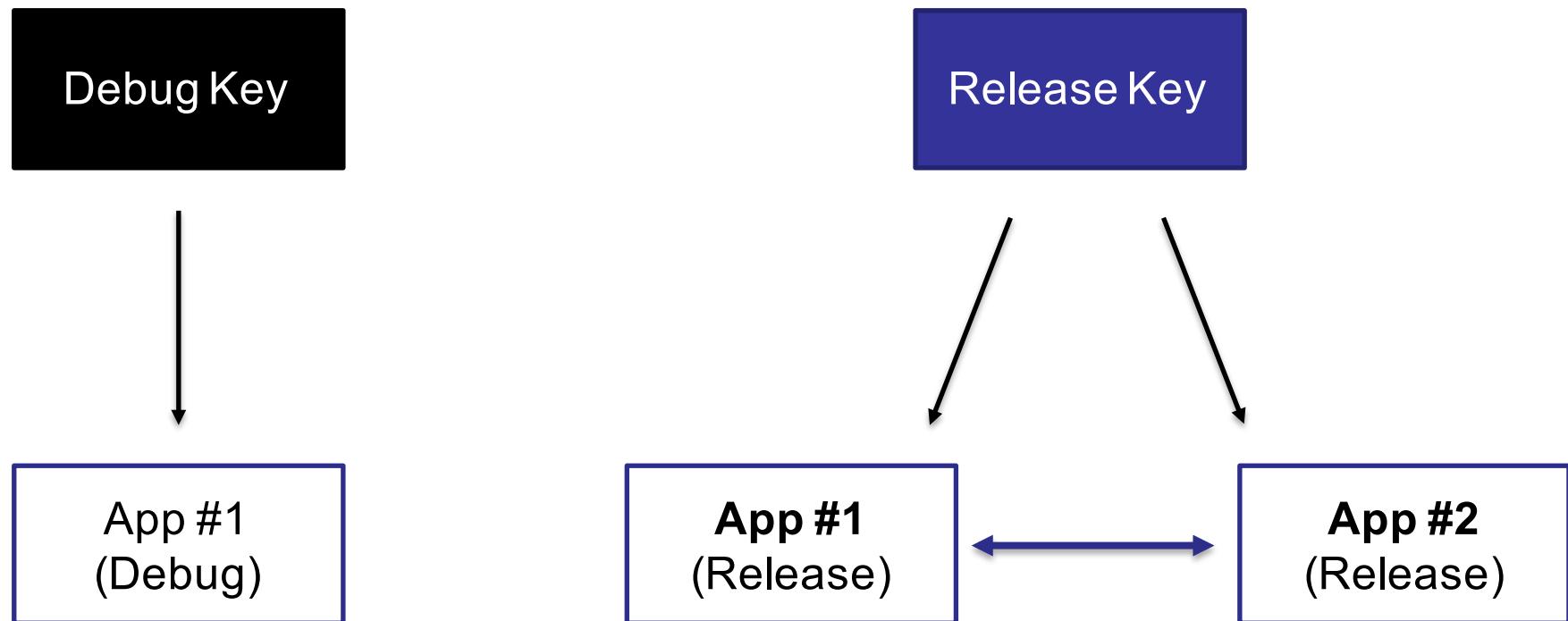
## List of “Dangerous” Permissions

- **CALENDAR** : READ\_CALENDAR, WRITE\_CALENDAR
- **CAMERA** : CAMERA CONTACTS : READ\_CONTACTS, WRITE\_CONTACTS, GET\_ACCOUNTS
- **LOCATION** : ACCESS\_FINE\_LOCATION, ACCESS\_COARSE\_LOCATION
- **MICROPHONE** : RECORD\_AUDIO
- **PHONE** : READ\_PHONE\_STATE, CALL\_PHONE, READ\_CALL\_LOG, WRITE\_CALL\_LOG, ADD\_VOICEMAIL, USE\_SIP, PROCESS\_OUTGOING\_CALLS
- **SENSORS** : BODY\_SENSORS
- **SMS** : SEND\_SMS, RECEIVE\_SMS, READ\_SMS, RECEIVE\_WAP\_PUSH, RECEIVE\_MMS
- **STORAGE** : READ\_EXTERNAL\_STORAGE, WRITE\_EXTERNAL\_STORAGE

Your apk must be signed to install

- Ensure the authenticity of the author on updates
- Establish trust relationship among apps signed with same key (Share Permission, UID, Process, Sandbox)

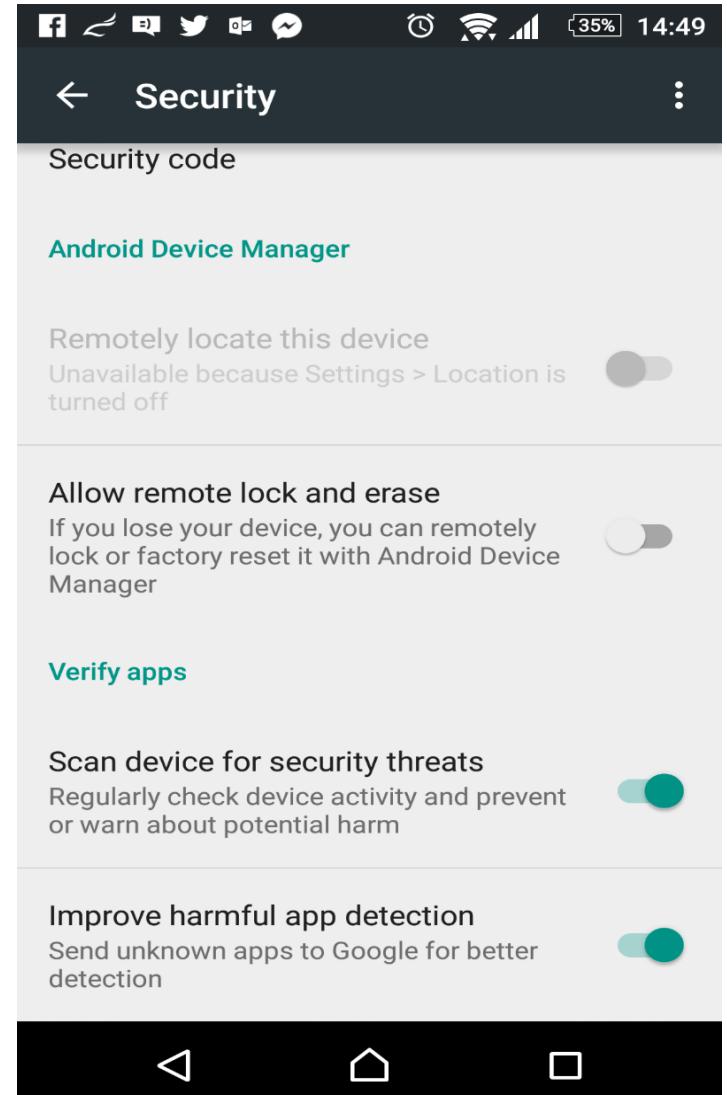
## Signing Type



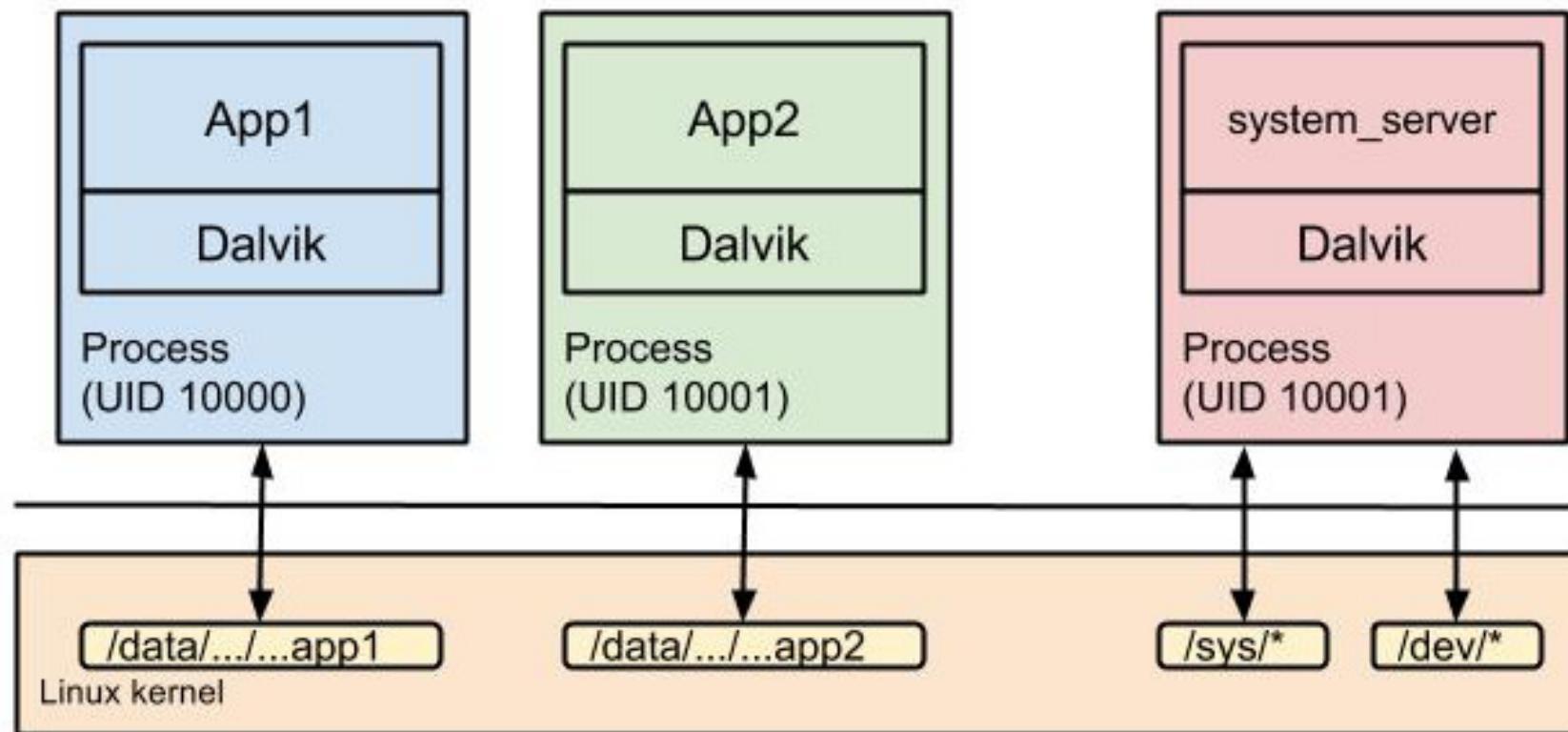
Same signing key means  
permissionLevel="signature" works!

## Risk mitigation for your Device

- Android 4.2 and later.
- Enable by default (Google Settings app).
- Alert when install harmful app or Block installation.



## Isolated File Storage





## Defend Customer Data (development tips)



Knowing IT

- Secure Client Storage
- Secure Transport Layer
- Reduce Information Leak
- Storing Secret Key
- Data Validation
- Secure Inter-process Communication
- Binary Protection

## 10 Million Customers At Risk

**[CVE-2014-0647] Insecure Data Storage of User Data Elements in Starbucks v2.6.1 iOS mobile application**

*From:* Daniel Wood <daniel.wood () owasp org>

*Date:* Mon, 13 Jan 2014 22:28:33 -0600

---

**Title:** [CVE-2014-0647] Insecure Data Storage of User Data Elements in Starbucks v2.6.1 iOS mobile application

**Published:** January 13, 2014

**Reported to Vendor:** December 2013 (no direct response)

**CVE Reference:** CVE-2014-0647

**Credit:** This issue was discovered by Daniel E. Wood

<http://www.linkedin.com/in/danielewood>

**Product:** Starbucks iOS mobile application

**Version:** 2.6.1 (May 02, 2013)

**Vendor:** Starbucks Coffee Company

**URL:** <https://itunes.apple.com/us/app/starbucks/id331177714>

**Issue:** Username, email address, and password elements are being stored in clear-text in the session.clslog crashlytics log file.

**Location:** /Library/Caches/com.crashlytics.data/com.starbucks.mystarbucks/session.clslog

Within session.clslog there are multiple instances of the storage of clear-text credentials that can be recovered and leveraged for unauthorized usage of a users account on the malicious users' own device or online at <https://www.starbucks.com/account/signin>. It contains the HTML of the mobile application page that performs the account login or account reset. session.clslog also contains the OAuth token (signed with HMAC-SHA1) and OAuth signature for the users account/device to the Starbucks service.

# Wine App webview.db plaintext

The screenshot shows the DB Browser for SQLite interface. The main window displays the 'password' table with the following data:

_id	host	username	password
1	httpsaccounts.google.com	likeavinash	[REDACTED]
2	httpsm.facebook.com	likeavinash	[REDACTED]

The 'DB Schema' panel on the right shows the database structure:

- Tables (5)
  - android\_metadata
  - locale
  - formdata
    - \_id
    - urlid
    - name
    - value
  - formurl
    - \_id
    - url
  - httpauth
    - \_id
    - host
    - realm
    - username
    - password
  - password
    - \_id
    - host
    - username
    - password- Indices (3)
  - sqlite\_autoindex\_formdata\_1
  - sqlite\_autoindex\_httpauth\_1
  - sqlite\_autoindex\_password\_1
- Views (0)
- Triggers (0)

At the bottom, there are buttons for SQL Log, Plot, and DB Schema, and the text 'UTF-8'.

## MODE\_WORLD\_READABLE

The screenshot shows an Android device's root shell via ADB. The terminal window title is "whitehatpanda — adb shell — adb shell — 96x25". The command `ls -laR` is run to list directory contents with detailed permissions. A red box highlights the file `MyInternalFile` in the `files` directory, which has world-readable permissions (`-rw-rw-r--`). Another red box highlights the password dump at the bottom, which includes the file path `com.hptsec.vulnlab/files/MyInternalFile`, the text "Android Internal Storage With World Readable", and the credentials "Username:admin" and "Password:P@ssw0rd".

```
root@android:/data/data/com.hptsec.vulnlab # ls -laR

.:
drwxrwx--x u0_a100 u0_a100 2015-11-06 07:31 cache
drwxrwx--x u0_a100 u0_a100 2015-11-06 07:33 files
drwxr-xr-x system system 2015-11-06 07:31 lib

./cache:
drwx----- u0_a100 u0_a100 2015-11-06 07:31 com.android.renderscript.cache

./cache/com.android.renderscript.cache:

./files:
-rw-rw-r-- u0_a100 u0_a100 78 2015-11-06 07:33 MyInternalFile

./lib:
root@android:/data/data/com.hptsec.vulnlab # su shell
shell@android:/data/data/com.hptsec.vulnlab $ id
uid=2000(shell) gid=2000(shell)
com.hptsec.vulnlab/files/MyInternalFile
Android Internal Storage With World Readable
Username:admin
Password:P@ssw0rd
shell@android:/data/data/com.hptsec.vulnlab $
```

## Development Tips

- Store data in app sandbox
- Encrypt high sensitive data in `shared_prefs`,  
`sqlite db`, etc
- Don't use `MODE_WORLD_READ/WRITEABLE`.
- API < 19, SD Card is readable by default

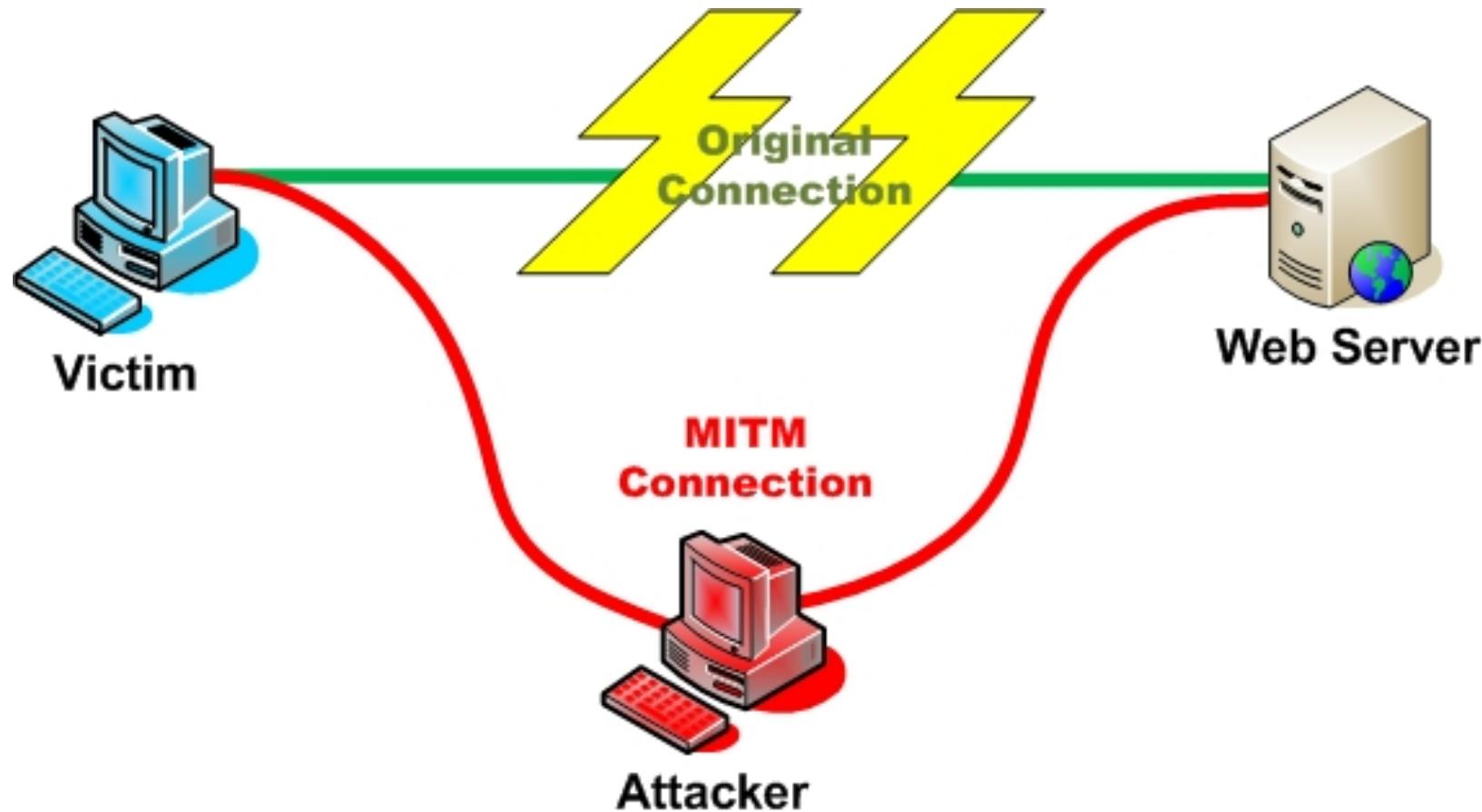
# Where is the Encryption Key?

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string
name="sdk_phone_number">iHCgspXVW8Xic4ZyRZ77RoGA=
=</string>
<string name="sdk_is_active">HAWA7OUMK1QI=</string>
<string
name="sdk_fullname">WX1f0crHrpQxKxq7Uk2L4WtEQ==</str
ing><string </map>
</xml>
```

There're too many un-trusted network  
'round me

- Authenticating user through HTTPS
- Certificate Pinning
- Review 3<sup>rd</sup>-party data transfer over HTTP
- Do not Trust all Certificates

## Man-in-The-Middle



## Various app fails to validate SSL

Android apps that fail to validate SSL

whitehat.panda@gmail.com ▾

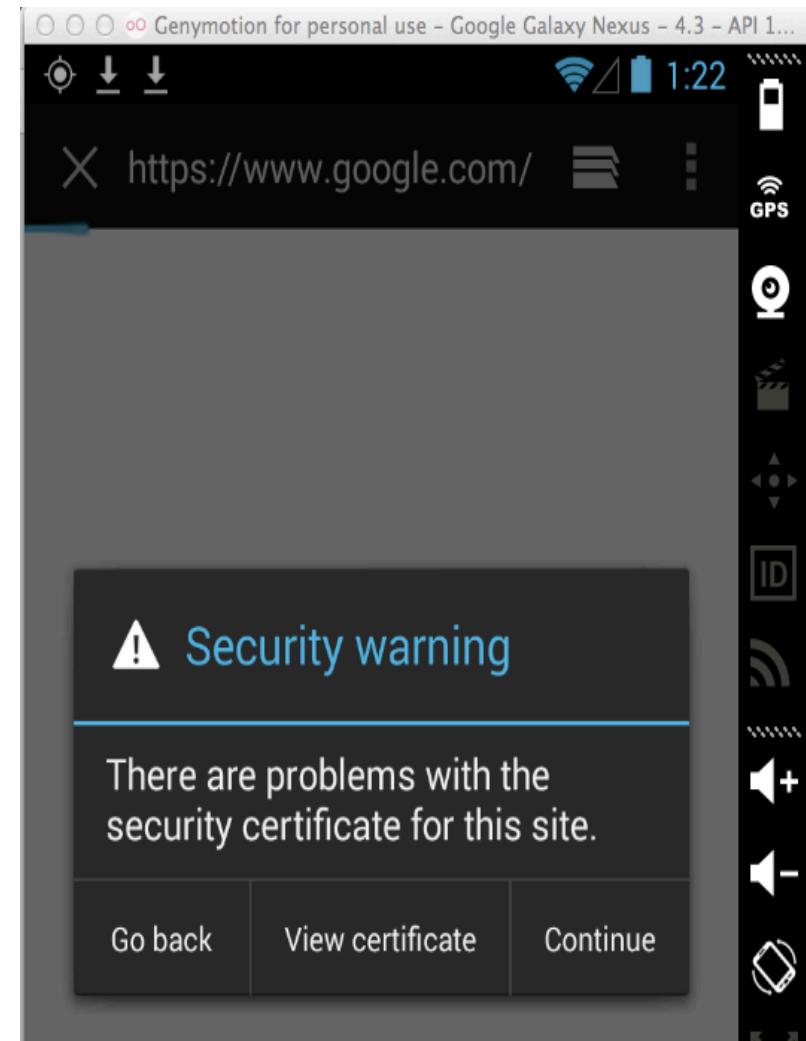
Share

Temporary filter 1 Range: A1:AA23668

	A	B	C	D	E	F	G	H	I	J	K
1	App	Link	Genre	Count	Date added	Version tested	malldroid broker	Library traffic observed	Non-library traffic observed	Verified	Vulnerabl
19470	Pence Wealth Management	<a href="#">com.mobileuploader.a</a>	Finance	50+	2014-10-21	219796	TRUE	TRUE	TRUE	Yes	Yes
20858	Huntington Mobile	<a href="#">com.huntington.m</a>	Finance	500,000+	2014-09-03	2.1.222	Maybe		TRUE	Yes	Yes
20860	State Bank Anywhere	<a href="#">com.sbi.SBIFreedomPl</a>	Finance	500,000+	2014-09-13	2.0.1	FALSE		TRUE	Yes	Yes
20862	Finansbank Cep Şubesi	<a href="#">com.finansbank.mobile</a>	Finance	500,000+	2014-09-16	1.1.5	TRUE		TRUE	Yes	Yes
20863	Альфа-Банк (Alfa-Bank)	<a href="#">ru.alfabank.mobile.andr</a>	Finance	500,000+	2014-09-25	5.5.1.1	TRUE		TRUE	Yes	Yes
20864	Westpac Mobile Banking	<a href="#">org.westpac.bank</a>	Finance	500,000+	2014-09-25	5.21	TRUE		TRUE	Yes	Yes
20866	Virtual Wallet by PNC	<a href="#">com.pnc.ecommerce.m</a>	Finance	500,000+	2014-10-01	2.1.1	TRUE		TRUE	Yes	Yes
20867	Supermóvil	<a href="#">mx.bancosantander.sup</a>	Finance	500,000+	2014-10-21	2.0	TRUE		TRUE	Yes	Yes
20965	Kmart	<a href="#">com.kmart.android</a>	Shopping	500,000+	2014-09-10	6.2.6	TRUE		TRUE	Yes	Yes
20966	Sears	<a href="#">com.sears.android</a>	Shopping	500,000+	2014-09-10	6.2.6	TRUE		TRUE	Yes	Yes
20971	MininTheBox Online Shopping	<a href="#">com.miniinthethebox.andro</a>	Shopping	500,000+	2014-09-15	2.0.0	Maybe	TRUE	TRUE	Yes	Yes
21708	United Educational CU	<a href="#">com.metova.cuae.uecu</a>	Finance	500+	2014-09-27	1.0.27	TRUE		TRUE	Yes	Yes
21709	United Advantage NW Federal Cr	<a href="#">com.myappengine.uan</a>	Finance	500+	2014-09-27	1.7	TRUE		TRUE	Yes	Yes
21711	Smart 어음정보	<a href="#">kr.or.knote.android</a>	Finance	500+	2014-10-02	1.0.3	FALSE		TRUE	Yes	Yes
21712	Quest Federal CU Mobile	<a href="#">com.metova.cuae.ques</a>	Finance	500+	2014-10-09	1.0.27	TRUE		TRUE	Yes	Yes
21713	Forest Area FCU Mobile	<a href="#">com.metova.cuae.fafcu</a>	Finance	500+	2014-10-10	1.0.29	TRUE		TRUE	Yes	Yes
22609	LocalSense	<a href="#">com.LocalSense</a>	Shopping	500+	2014-09-28	1.2.1	FALSE		TRUE	Yes	Yes

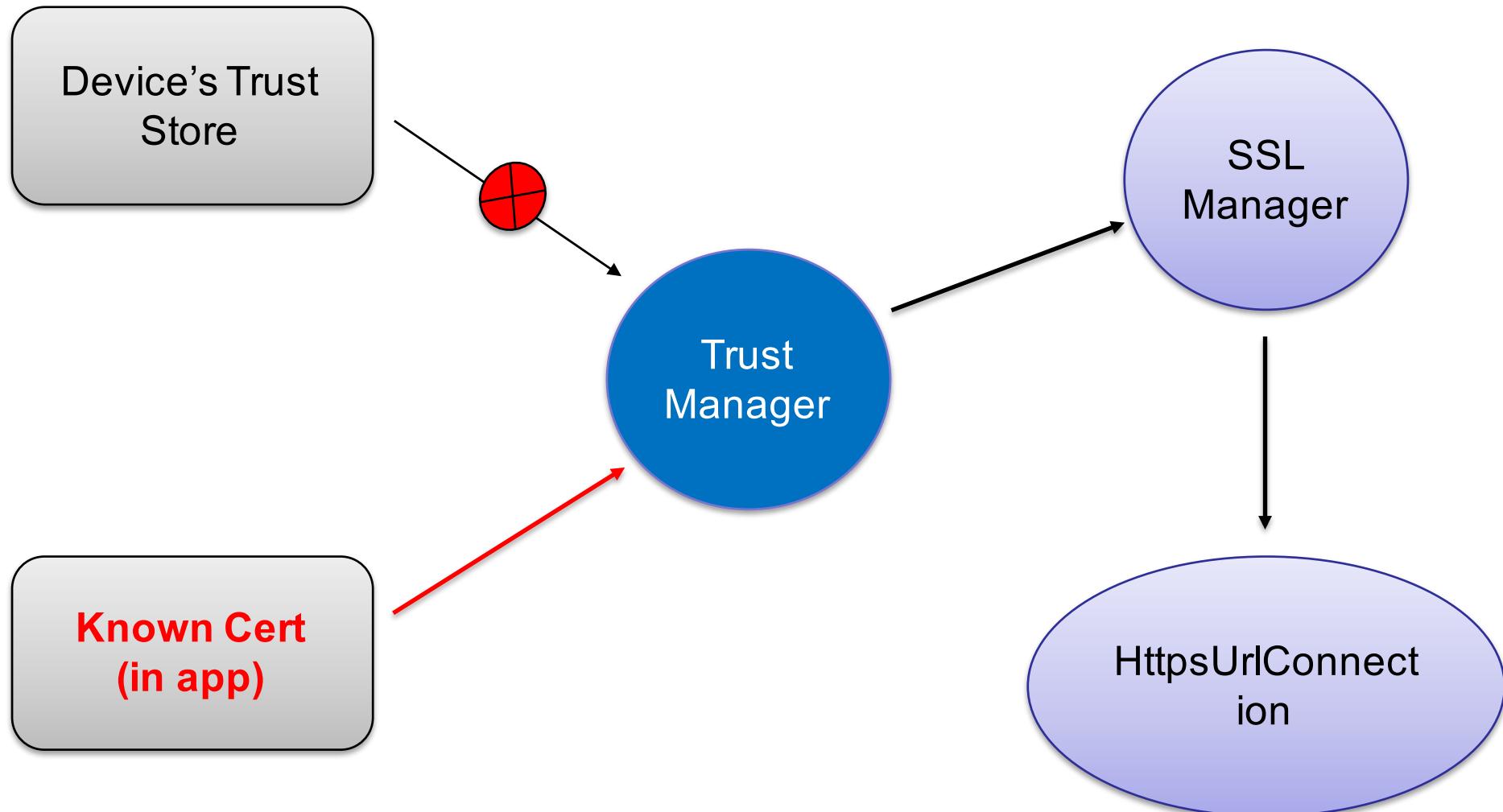
## Common Mistakes

- Do not check Server certificates
- Do not verify hostname
- Ignore SSL errors

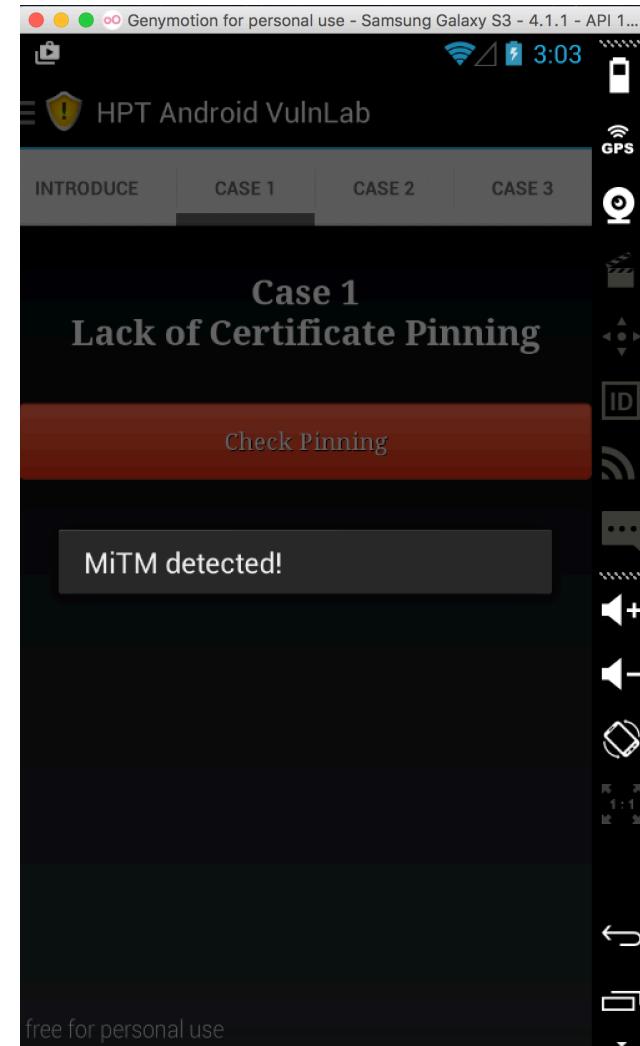
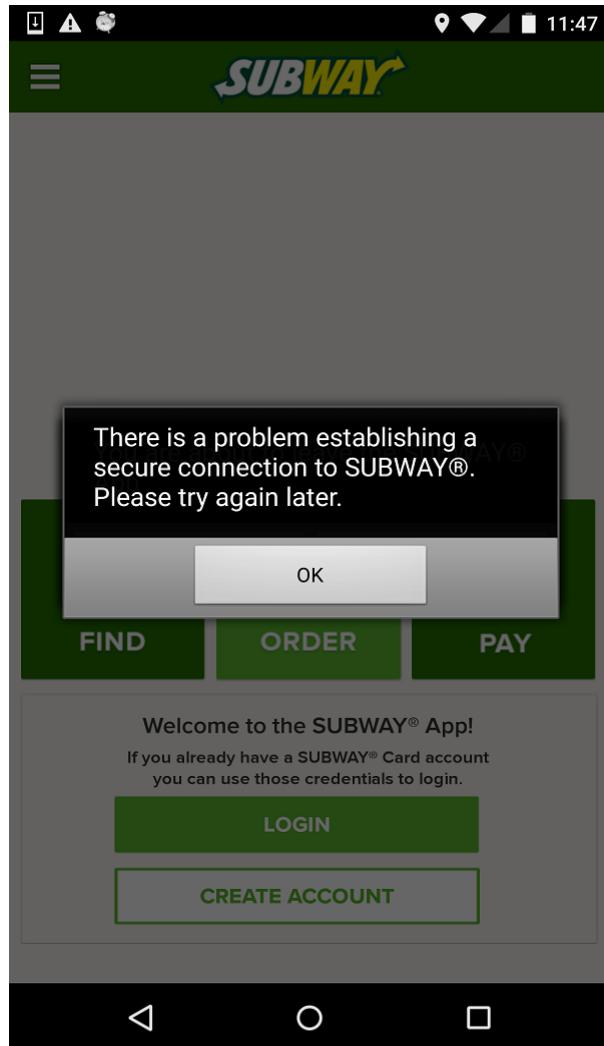


# Secure Transport Layer

Certificate Pinning – Only my Cert



## Customers have more belief in Application



# Certificate Pinning Demo

Sublime Text File Edit Selection Find View Goto Tools Project Window Help Certificate Pinning Demo — BugBounty, rop UNREGISTERED

```
< > Certificate Pinning Demo
1 Certificate Pinning Demo
2
3 + When no one between you and server
4 + When you are in the MiTM condition (through a SSL proxy like
Burp Suite)
```

24 Words, Line 4, Column 11 Tab Size: 4 Plain Text

A horizontal row of small, colorful icons representing various Mac OS X applications, including Finder, Mail, Safari, and others.

# Reduce Information Leak

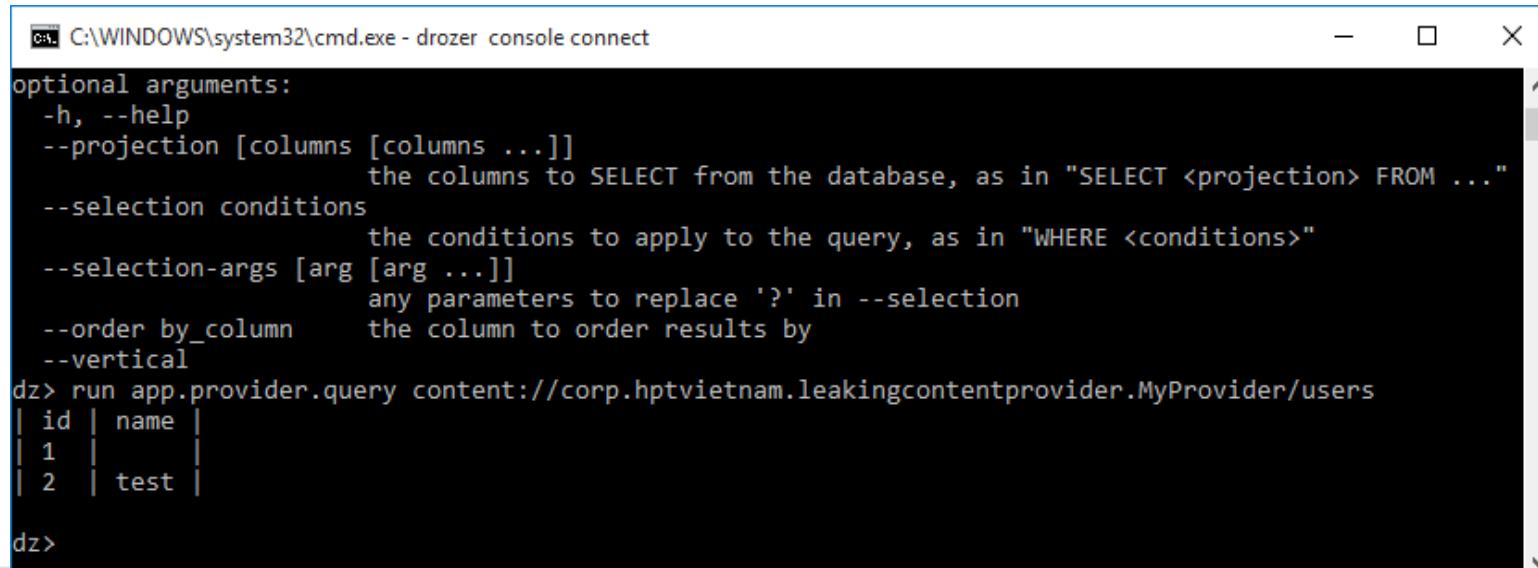
## Android Logcat – System Log

```
public void login(String userid, String password, OnLoginCompleted callback) throws
Exception {
    StringBuffer v1 = new StringBuffer("");
    v1.append("{");
    v1.append(String.format("\"app_key\":\"%s\"", "xxxxxxxx"));
    v1.append(String.format(",\"user_email\":\"%s\"", userid));
    v1.append(String.format(",\"password\":\"%s\"", password));
    v1.append("}");
    HttpRequest vo = new HttpRequest();
    vo.setUrl(new URL("http://xxx.xxx/login"));
    vo.setBody(v1.toString());
    Log.e("LOGIN_DATA", v1.toString());
    new APICallTask(this.client.getHttpUtil(), new OnAPICallCompleted() {
        ...[SNIP]...
    }).execute(new Object[]{vo});
}
```

# Reduce Information Leak

## Secure Content Provider

```
<provider
    android:name=".MyProvider"
    android:authorities="corp.hptvietnam.leakingcontentprovider.MyProvider"
    android:exported="true"
</provider> (API <= 16: exported true by default)
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - drozer console connect". The window displays the help documentation for the "run" command of the drozer tool, which is used for performing database queries. The help text includes options for projection, selection conditions, and arguments. Below the help text, a database query is run against a ContentProvider named "corp.hptvietnam.leakingcontentprovider.MyProvider" with the path "/users". The query returns two rows of data: one with id 1 and name "test", and another with id 2 and name "test".

```
optional arguments:
-h, --help
--projection [columns [columns ...]]
    the columns to SELECT from the database, as in "SELECT <projection> FROM ..."
--selection conditions
    the conditions to apply to the query, as in "WHERE <conditions>"
--selection-args [arg [arg ...]]
    any parameters to replace '?' in --selection
--order_by_column    the column to order results by
--vertical
dz> run app.provider.query content://corp.hptvietnam.leakingcontentprovider.MyProvider/users
| id | name |
| 1  | test |
| 2  | test |
dz>
```

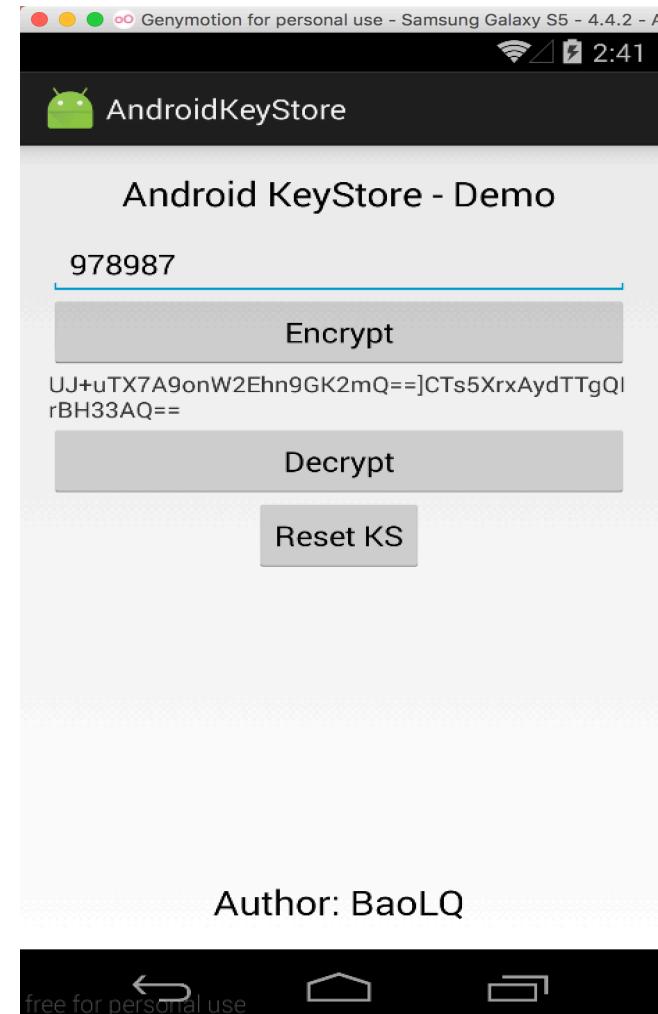
## Hardcoded Key is Bad

- It is very easy to figure out

```
static {  
    MyAESKey.value = new byte[]{84, 104, 101, 66,  
    101, 115, 116, 83, 101, 99, 114, 101, 116, 75, 101, 121};  
}
```

# Are there any way to store EncKey safety?

- Android Keystore is the best way
- Support AES – CBC, RSA (private/public key)...
- More complex than the old one



<https://github.com/nelenkov/android-keystore>

```
Sublime Text  File  Edit  Selection  Find  View  Goto  Tools  Project  Window  Help  Android KeyStore Demo — BugBounty, rop  Fri 11:22  UNREGISTERED
< > Android KeyStore Demo
1 Android KeyStore Demo
2
3 + The best way to store encryption key on unrooted device (cause
of there is no security boundary in rooted one)
4
5 + Device lock screen PIN, Password must be set
6
7 + Harder to implement than hardcoded value
8
9 => But there is an open source project can save our time (credit
to the author)
10
11 References:
12 + https://bitbucket.org/baolq/androidkeystore
13 + Credit to: https://github.com/nelenkov/android-keystore (it's
also supporting for more encryption method)

62 Words, Line 3, Column 12  Tab Size: 4  Plain Text

```

## Parameterized Statement

- The oldest bug still alive in SQLite DB

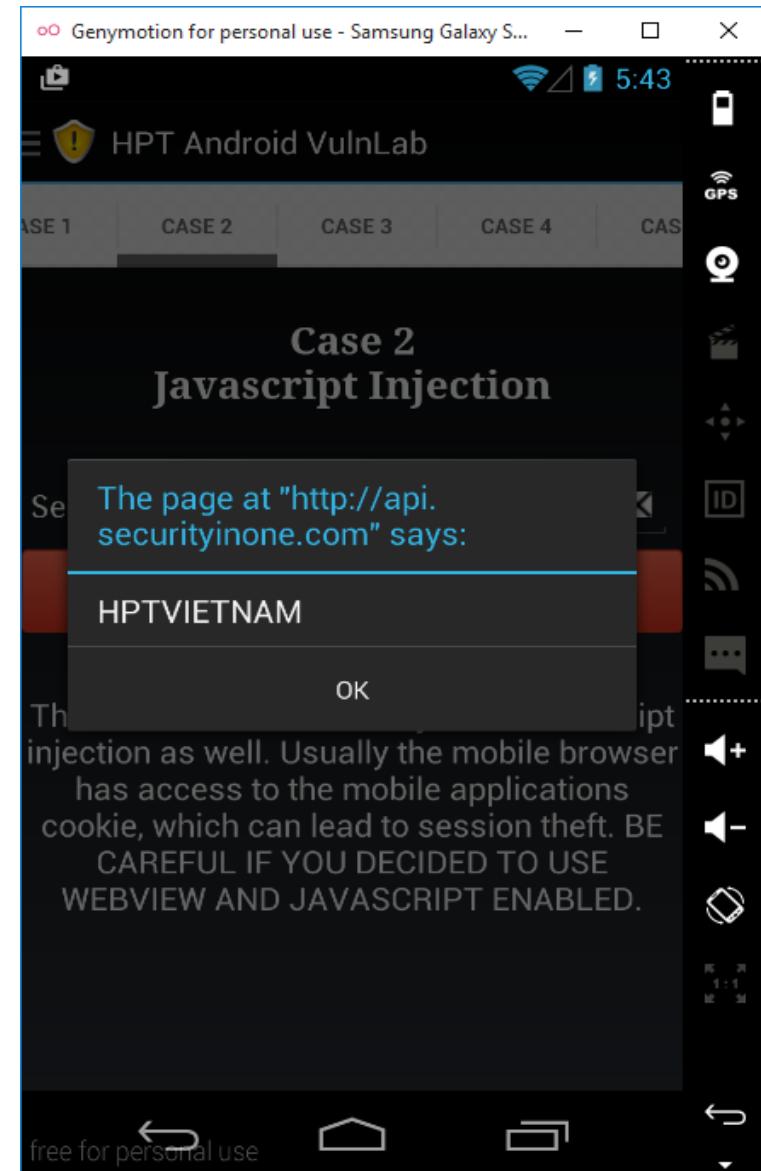
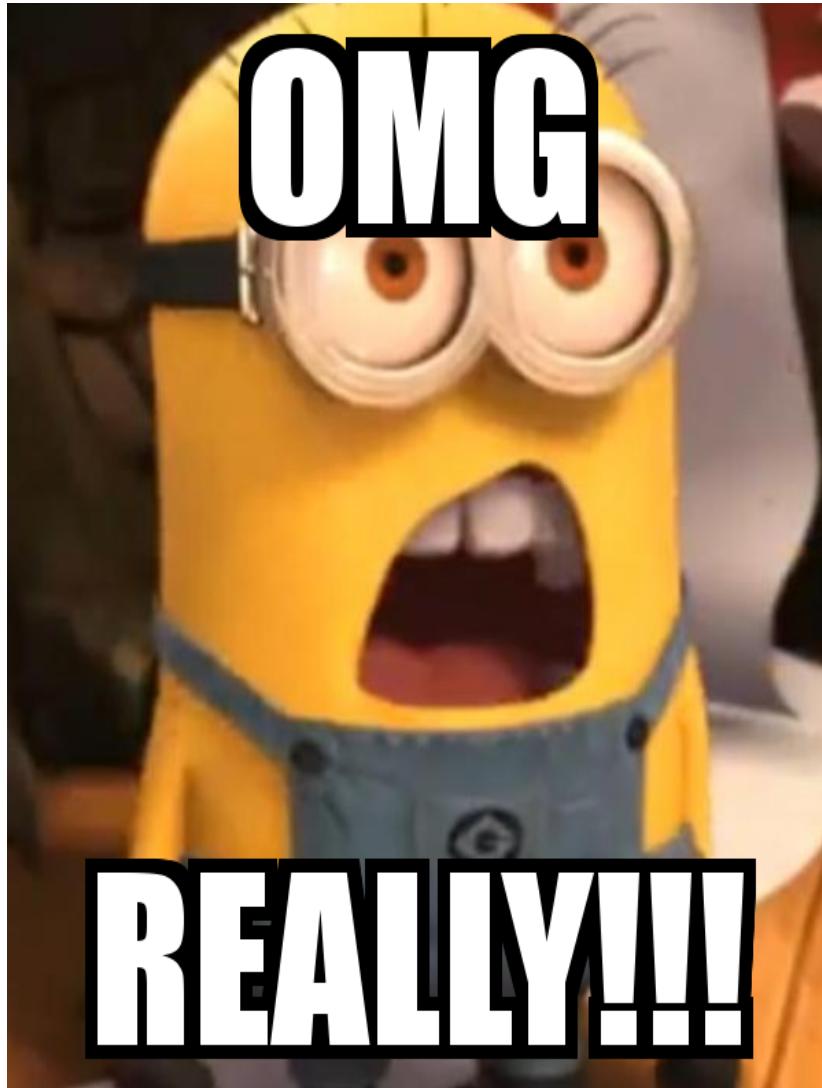
```
query = "SELECT * FROM " + TABLE_NOTE + " WHERE " +  
KEY_NOTEDATA + " LIKE '%" + mySearch + "%' AND " +  
KEY_PRIVATENOTE + " = 1";
```

```
query = "SELECT * FROM TABLE_NOTE WHERE  
KEY_NOTEDATE LIKE ?";  
PreparedStatement prepStmt =  
con.prepareStatement(selectStatement);  
prepStmt.setString(1, "%" + mySearch + "%");
```

## Disable WebView Javascript/FileAccess

- Disable JavaScript or validate the input on server

```
WebSettings webSetting = webView.getSettings();
webSetting.setJavaScriptEnabled(false);
webSetting.setAllowFileAccess(false);
webView.setWebChromeClient(new WebChromeClient());
webView.loadUrl(url);
```



## Sanitize input URI on client

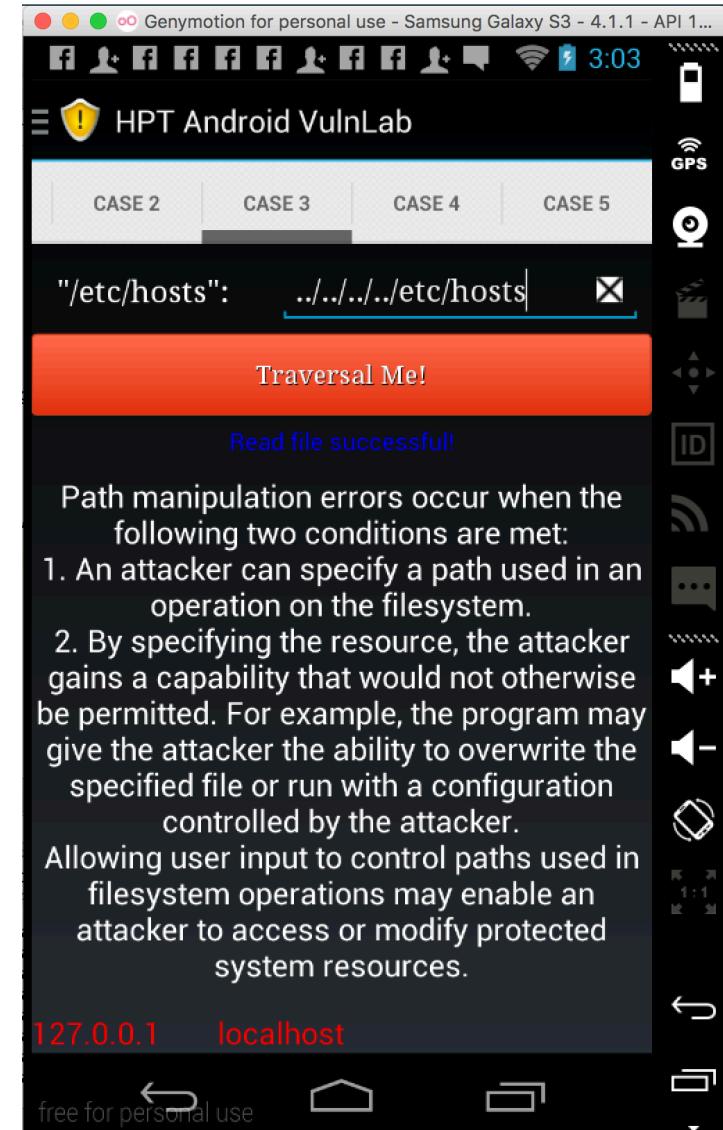
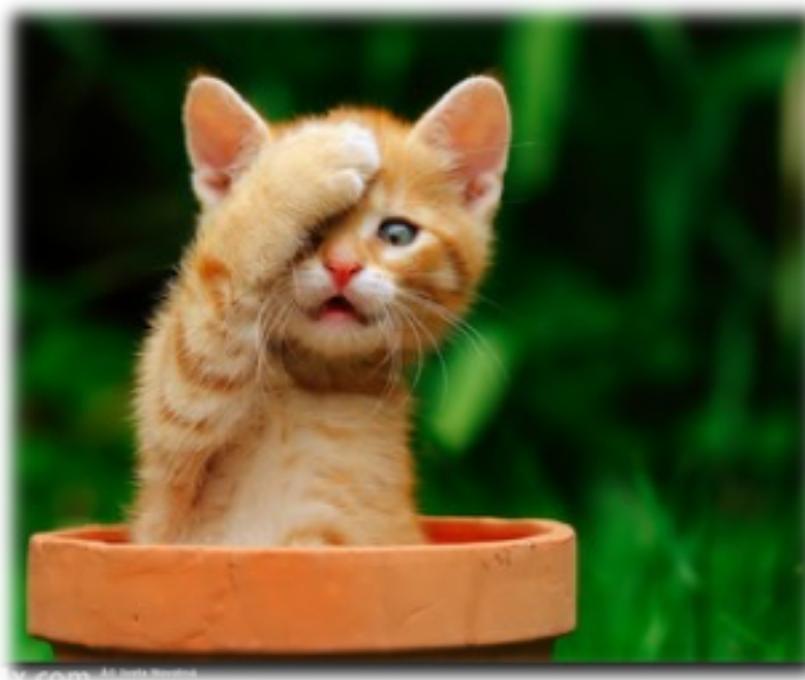
- Exported content provider
- Don't sanitize any untrusted URI input

```
# Pass malicious path to FileHandle  
ParcelFileDescriptor myFilePointer =  
getActivity().getContentResolver().openFileDescriptor(Uri.parse  
(FileHandleProvider.PROVIDER_URI.toString() + myUri), "r");
```

```
# FileHandle -> Open File  
File f = new File(getContext().getFilesDir(), fullUri.getPath());  
if (f.exists()) { return ParcelFileDescriptor.open(f,  
ParcelFileDescriptor.MODE_READ_ONLY); }
```

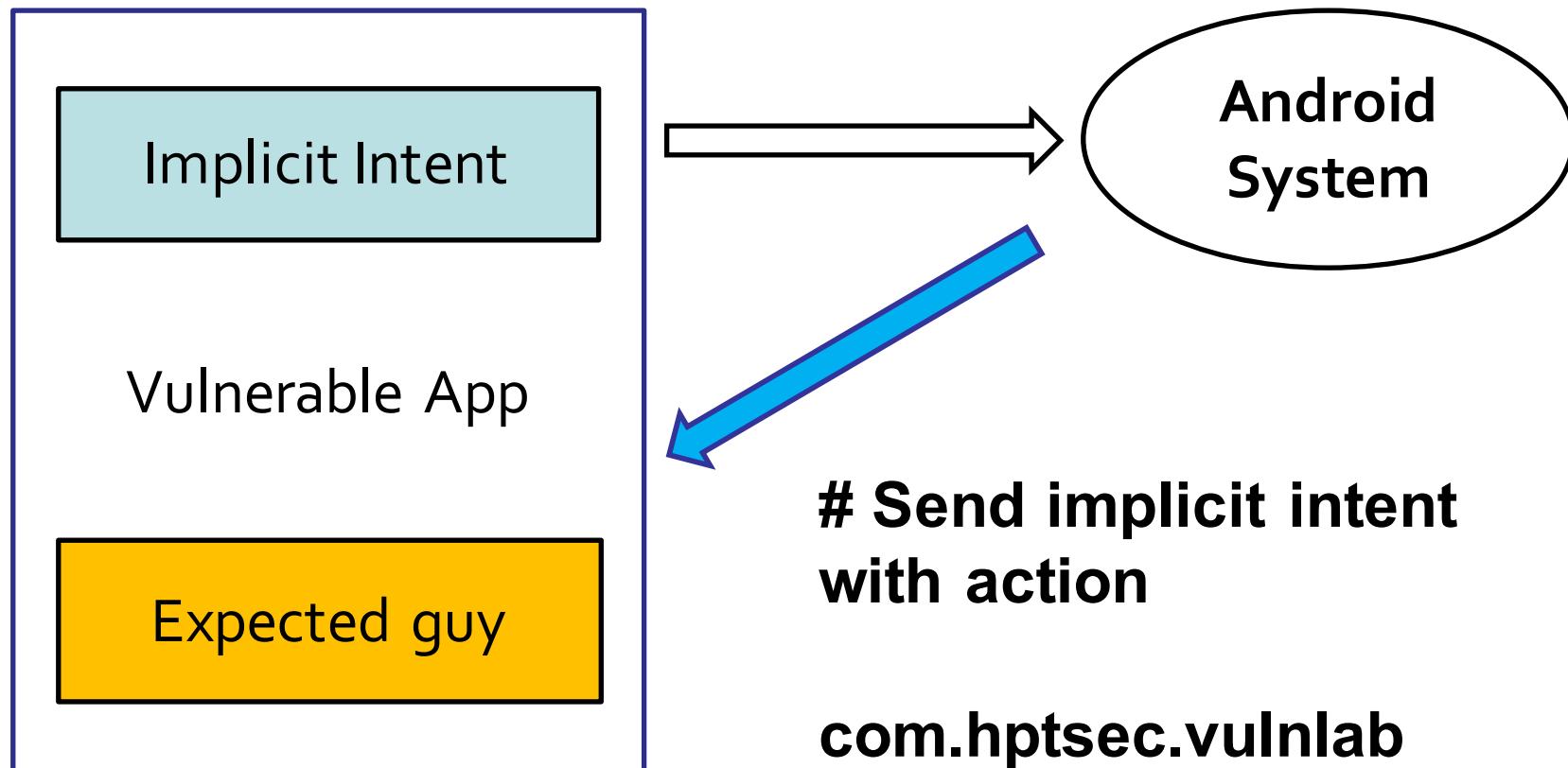
## Path Traversal - PoC

Aww! Noo. Pray for him!



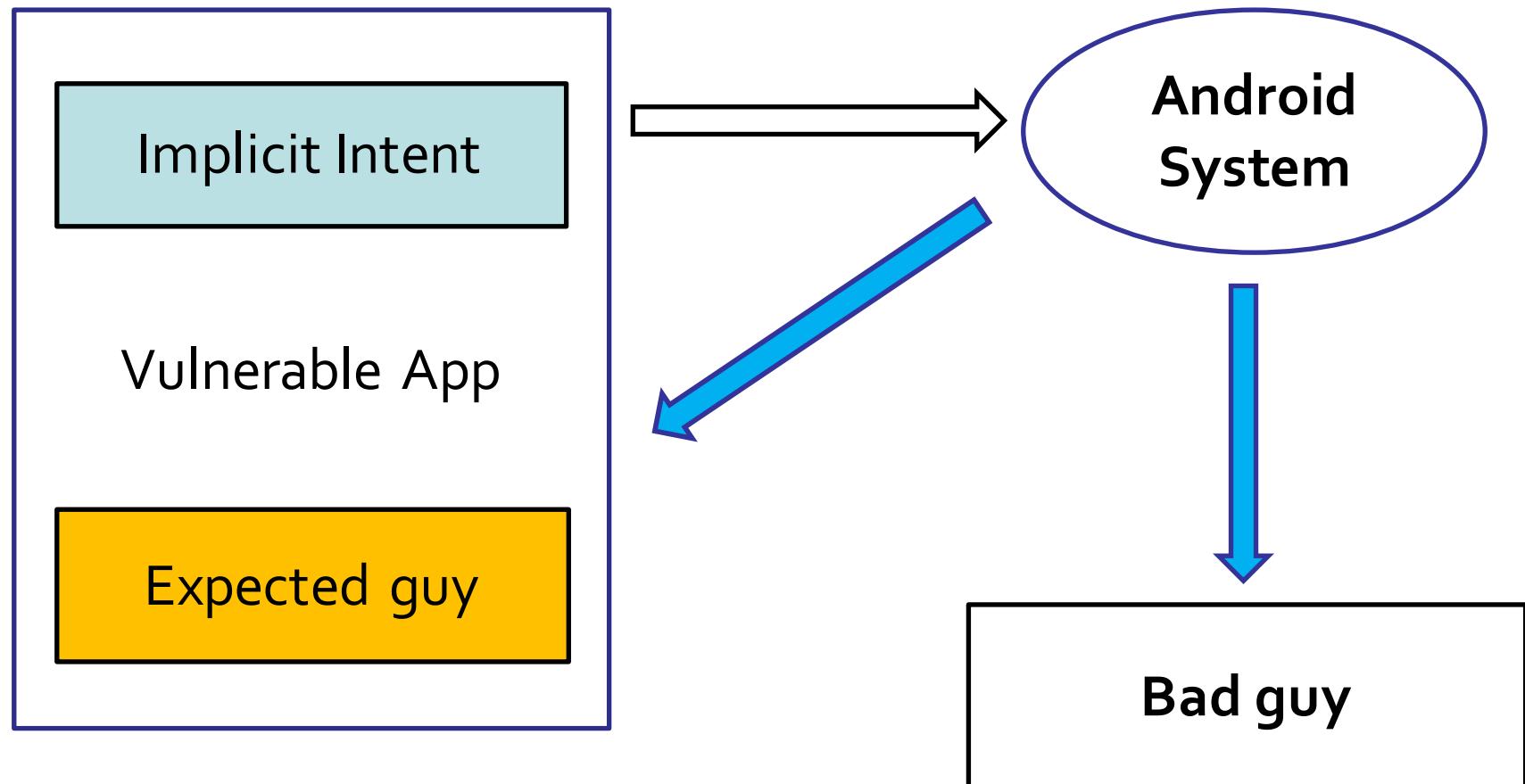
## When implicit Intent call

- Case #1

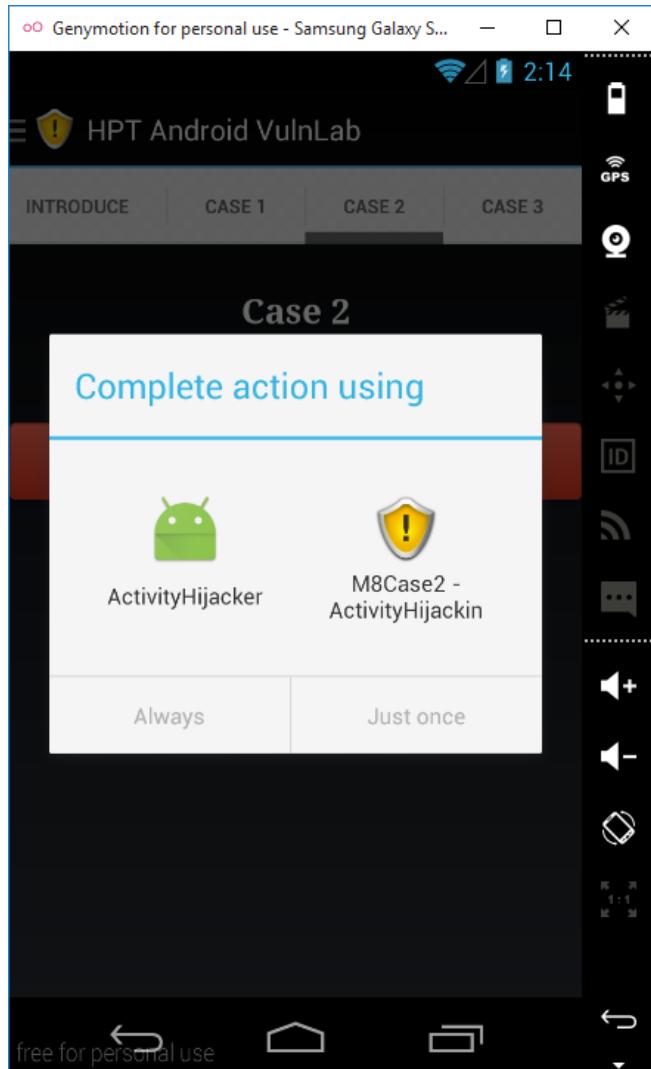


End-user doesn't know about this!

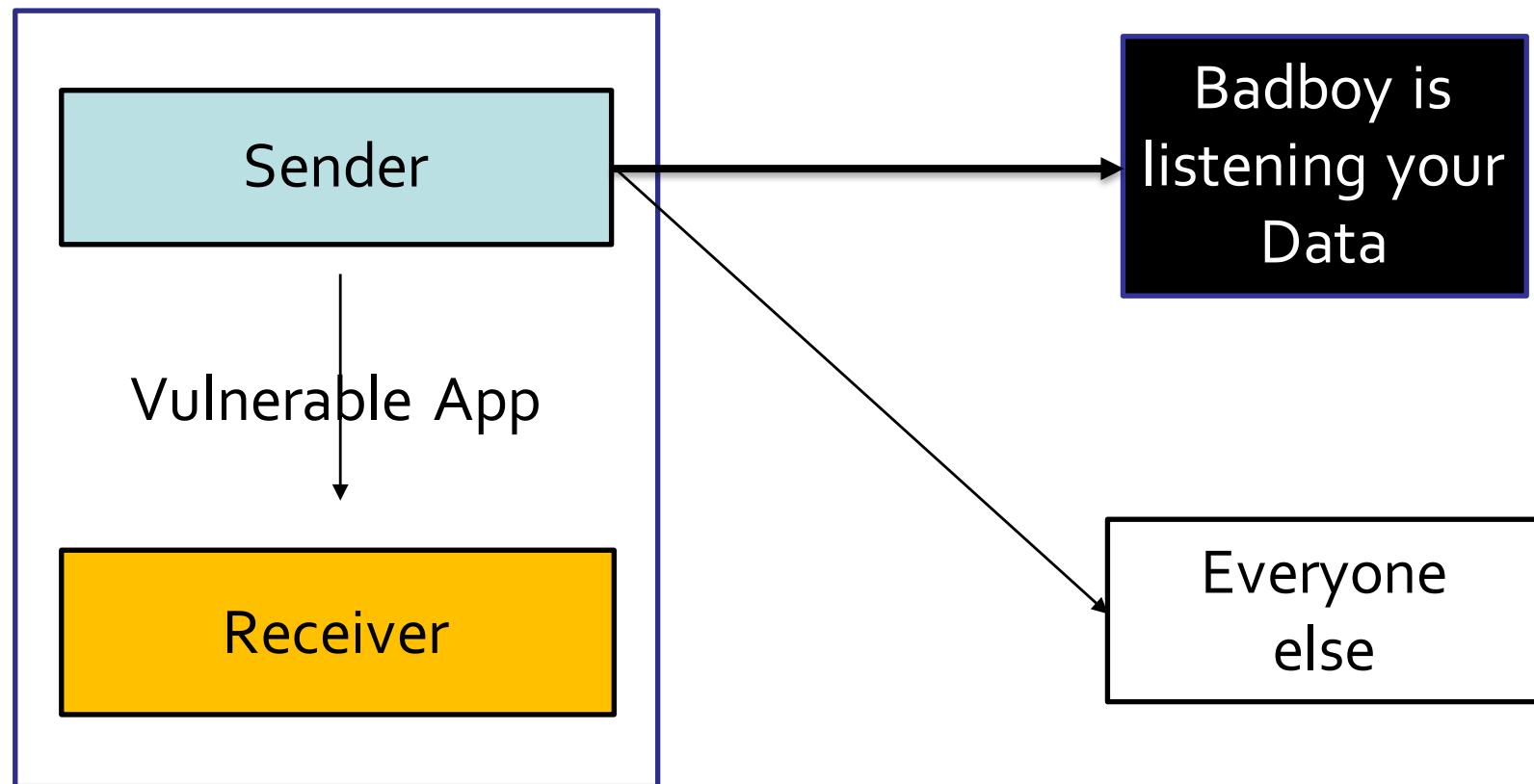
- Case #2



## At the street corner?



Don't put sensitive info in broadcast intent



Sublime Text File Edit Selection Find View Goto Tools Project Window Help

Broadcast Call without Permission — BugBounty, rop UNREGISTERED

< > Broadcast Call without Permission

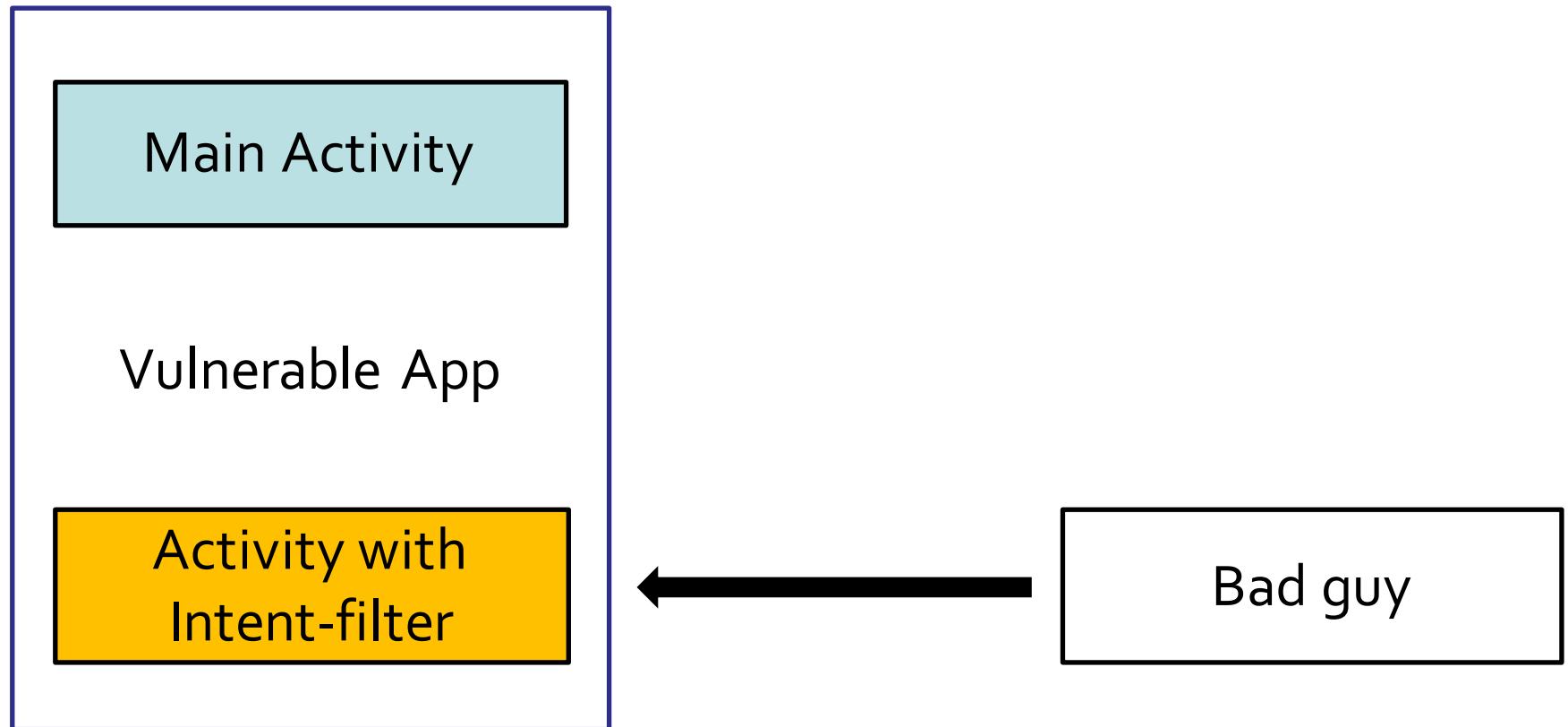
```
1 Broadcast Call without Permission
2
3 + Attacker: no permission required
4 + Victim: CALL_PHONE, exported via Intent-filter
5
6
7
```

12 Words, Line 6, Column 1 Tab Size: 4 Plain Text

A horizontal row of Mac OS X Dock icons, including Finder, Mail, Safari, and various system and application icons.

## Secure Activity/Service

- Intent-filter will public every components use it



## Best Practices for IPC

- Define your own custom permission with “Signature” level
  - Apply it to each component in your AndroidManifest
  - Intent-filter is not security boundary
- The day hacker went away

## Protection Level = “signature”



The screenshot shows the AndroidManifest.xml file in the Android Studio editor. The code is annotated with yellow highlights to illustrate the protection level configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="corp.hptvietnam.secureactivity" >

    <!-- Step 1: Define Permission with <permission> tag with protectionLevel="signature" -->
    <permission android:name="corp.hptvietnam.secureactivity.TrustViewer" android:protectionLevel="signature" ></permission>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" ...>

            <!-- Define secret activity with intent-filter (implicit exported), but protected with android:permission -->
            <activity android:name=".SecretActivity" android:permission="corp.hptvietnam.secureactivity.TrustViewer" > <!-- android:exported="true" -->
                <intent-filter>
                    <action android:name="corp.hptvietnam.secureactivity.VIEW_SECRET"></action>
                    <category android:name="ANDROID.INTENT.CATEGORY.DEFAULT"></category>
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

## No permission then go way



```
whitehatpanda — adb logcat — adb logcat — 80x24
adb
adb logcat +
```

```
W/System.err(13892): java.lang.SecurityException: Permission Denial: starting Intent { act=corp.hptvietnam.secureactivity.VIEW_SECRET cmp=corp.hptvietnam.secureactivity/.SecretActivity } from ProcessRecord{538d7784 13892:corp.hptvietnam.callsecretactivity/u0a102} (pid=13892, uid=10102) requires corp.hptvietnam.secureactivity.TrustViewer
W/System.err(13892):     at android.os.Parcel.readException(Parcel.java:1425)
W/System.err(13892):     .java:1379)
W/System.err(13892): Activity(Activity.
W/System.err(13892): activity(Instrum
W/System.err(13892): result(Activity.
W/System.err(13892): result(Activity.
W/System.err(13892): y.startActivity
ForResult(FragmentActivity.java:839)
W/System.err(13892):     at android.app.Activity.startActivity(Activity.java:3522
)
W/System.err(13892):     at android.app.Activity.startActivity(Activity.java:3490
)
W/System.err(13892):     at corp.hptvietnam.callsecretactivity.MainActivity$1.onClick(MainActivity.java:33)
W/System.err(13892):     at android.view.View.performClick(View.java:4084)
```

Sublime Text File Edit Selection Find View Goto Tools Project Window Help

Secure Inter-Process Communication — BugBounty, rop UNREGISTERED

```
< > Secure Inter-Process Communication
1 Secure Inter-Process Communation
2
3
4 Scenario #1 (diff signature, no permission):
5     -> Can't start activity from external
6
7 Scenario #2 (diff signature, same permission):
8     -> Same as the previous one
9
10 Scenario #3 (same signature, same permission):
11     -> Signature matched
```

32 Words, Line 7, Column 26 Tab Size: 4 Plain Text

A horizontal row of small icons representing various Mac OS X applications, including Finder, Mail, Safari, and others.

## IPC Protection Summary

Type	Concepts
Private	<ul style="list-style-type: none"><li>• Exported=false</li><li>• Explicit intent call (by class name)</li><li>• Validate data on send &amp; receiver context</li><li>• No Intent-Filter (if exists, enforce permission)</li></ul>
Public	<ul style="list-style-type: none"><li>• Exported=true</li><li>• Validate every Intent's Extra on receive</li><li>• Do not perform critical action based on these info</li></ul>
In-house	<ul style="list-style-type: none"><li>• Exported=true or via Intent-Filter</li><li>• Use custom permission with protectionLevel="Signature"</li><li>• Enforce permission for each component</li></ul>

- Code Obfuscation
- Integrity Checksum
- Release Config

- Enable proguard in release mode

```
signingConfigs {  
    releaseConfig {  
        storeFile file("C:\\\\.android\\\\your.keystore")  
        storePassword "android"  
        keyAlias "androiddebugkey"  
        keyPassword "android"  
    } }  
buildTypes {  
    release {  
        minifyEnabled true  
        proguardFile 'proguard-android.txt'  
        signingConfig signingConfigs.releaseConfig  
    } }
```

## Integrity Checksum

- Almost application doesn't know it was repacked or not

```
pkInfo =  
mContext.getPackageManager().getPackageInfo(mContext.getPackageName(), PackageManager.GET_SIGNATURES);  
for (Signature signature : pkInfo.signatures) {  
    MessageDigest md = MessageDigest.getInstance("SHA");  
    md.update(signature.toByteArray());  
    result = Base64.encodeToString(md.digest(), Base64.DEFAULT);  
}
```

# Anti Repack Demo

Sublime Text File Edit Selection Find View Goto Tools Project Window Help Anti Repacking Demo — BugBounty, rop UNREGISTERED

```
1 Anti Repacking Demo
2
3 Scenario #1 (orignin):
4     + App #1: Correct Signature (it works!)
5
6 Scenario #2 (repack):
7     + Repack & Resign new apk
8     + Check result
```

21 Words, Line 6, Column 22 Tab Size: 4 Plain Text

A horizontal row of small, colorful icons representing various Mac OS X applications like Mail, Finder, Safari, and iPhoto.

# Release Config - Make it harder to Exploit!

- Something to do before publish your app
  - `Android:debuggable="false"`
  - `Android:allowBackup="false"`

## When debuggable = False

The screenshot shows an Android device's adb shell terminal window titled "whitehatpanda — adb shell — adb shell — 80x14". The terminal displays a list of installed packages:

```
adb
package:com.xcelenzy.cydia
package:corp.hptvietnam.callsecretactivity
package:corp.hptvietnam.secureactivity
package:eu.chainfire.supersu
package:hotspotshield.android.vpn
package:jp.co.omronsoft.openwnn
package:myGame.system
package:org.proxydroid
package:org.sbtools.gamehack
package:than.chi.mong.vn.game.mobo
package:vn.sohagame.fallentitan
```

At the bottom, a command is being run:

```
[root@android:/ # run-as corp.hptvietnam.secureactivity
run-as: Package 'corp.hptvietnam.secureactivity' is not debuggable
1|root@android:/ # ]
```

## Developer documentation on security

- **Presentation & Documents:**

<https://github.com/whitehatpanda/VN-SecurityDay-2015>

- ✓ For references

- **Android Secure Coding - EN:**

[http://www.jssec.org/dl/android\\_securecoding\\_en.pdf](http://www.jssec.org/dl/android_securecoding_en.pdf)

- ✓ Describes how various secure coding way to prevent hacker's eyes

- **Security and Permissions:**

<http://developer.android.com/guide/topics/security/permissions.html>

- ✓ SDK documentation on the Android permission system



Q & A

A decorative graphic element consisting of a grid of small, light-gray squares arranged in a diamond pattern, with a larger red rectangular area overlaid containing the text "Q & A".

knowing IT

A large, semi-transparent gray text block containing the words "knowing IT".



Thank You

knowing IT