Advanced Android Malware Detection Framework

Xu Hao & pLL

2013.11

Content

- Background
- aDFAer - Android Data Flow Analyzer
- Janus - Detect Reflection
- Experiment
- Future Work

Android Security Problem

- APPs of Android
  - Android reaches massive 80% market share - second quarter of 2013
  - Malware threats growing at a rapid rate of 614 percent to 276,259 total malicious apps
    - Juniper Networks report
  - Android has become today's largest target for malware attacks
    - Sophos security threat report

Android Security Problem

- Google's dilemma
  - Google lost control of app auditing
    - More than 2/3 Apps are downloaded from third party Market rather than from Google Play
  - Google lost control of Android OS
    - Even google have good model to solve the serious problem, it's hard for it to put the model into effect

Android Malware

- Google for keyword: malware + android

Android-Based Spam Attack: A Smartphone Botnet In Action?

By Ken Presti

?? 05, 2012 7:59 ?? ET

A purported botnet is targeting Android-based smartphones as a means of delivering spam. The exploit leverages the Yahoo mail accounts of the phones' owners, and it is believed by some to be the first time that malware authors have managed to assemble an army of Ar

This deve

RECENT ARTICLES

The 10 Biggest Security Stories Of 2012

We review the highlights, as well as the low lights, of

nt of view. New

nly

a close but

Android Malware Creates Smartphone Botnet, Researchers Say

By Jeffrey Burt | Posted 2012-07-05 Email Print

LinkedIn 0 Twitter 0 Facebook 0 Share 0

RELAT

- DataMc
  Email E

In one year, Android malware up 580%, 23 of the top 500 apps on Google Play deemed 'High Risk'

Existent Analysis Technique

- Signature based
  - Traditional antivirus
- Host based
  - LBE, 360 Safe, NetQin, ...
- Dynamic taint analysis
  - DroidBox, TaintDroid, SmartDroid, ...
- Static taint analysis
  - FlowDroid, AndroidLeaks, SCANDAL, ...

Advanced Malware Technique

- To hide malicious operation
  - Using reflection mechanism
    - Invoke method at runtime to escape static scanning
- Dynamic code loading
  - Be able to load another DEX/jar file from local or network

Reflection Mechanism

- Reflection allows APP to create a method pointer at runtime and invoke it
- Malware may get pointer of some sensitive methods like “sendTextMessage” and store it for later use
- Hard to determine if sensitive method is called by simply static scanning
- Detect reflection is easy, BUT some benign APPs may use reflection

Dynamic Code Loading

- Pretend to be a normal APP
  - Download malware part and execute it under certain condition
  - Hide malware part from static analysis
  - Bypass APP audit
- dalvik.system.DexClassLoader
  - public DexClassLoader (String dexPath, String optimizedDirectory, String libraryPath, ClassLoader parent)

Sample Code

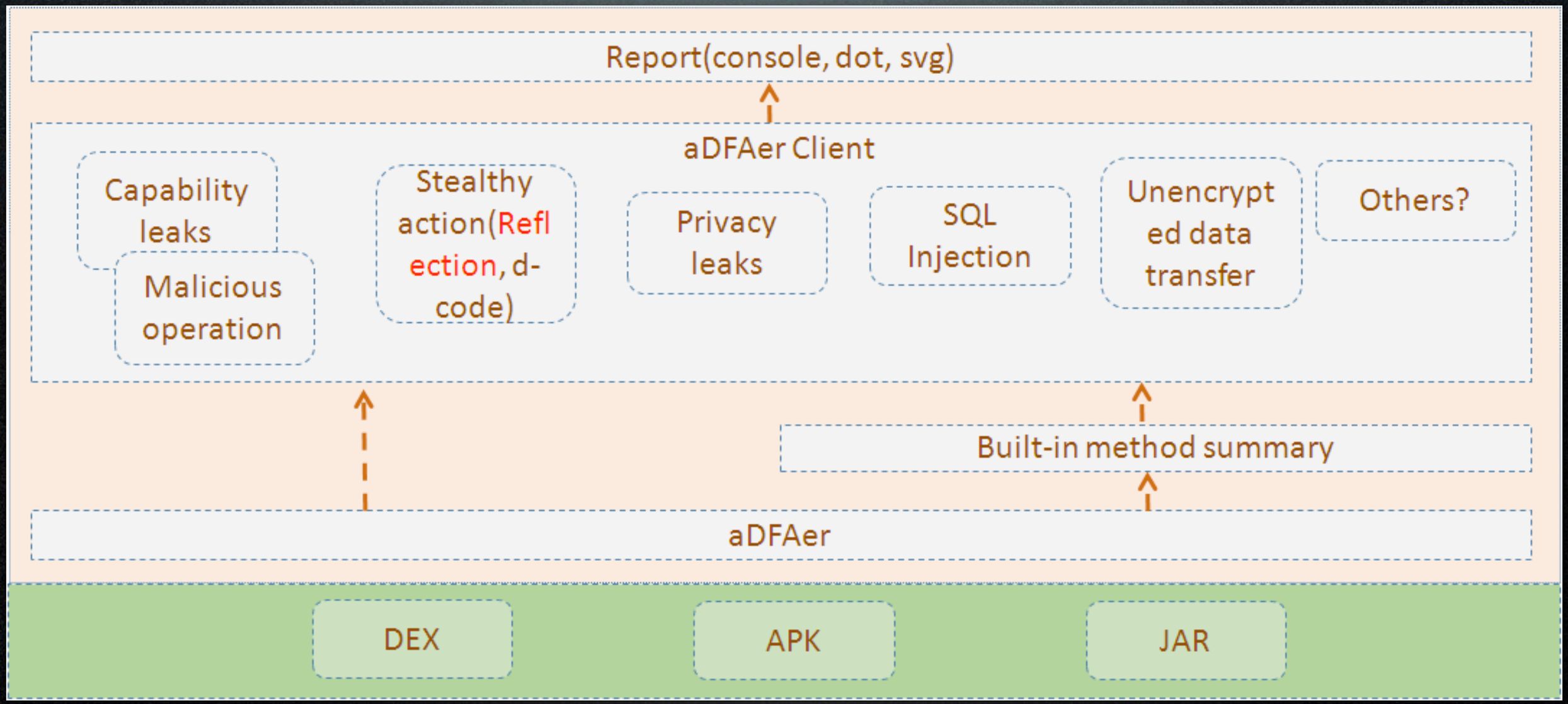
- Download an APK from network
- Dynamic load the APK file
- Invoke malicious method by reflection

```
// load APK we just download
File file = new File(context.getDir("temp", 0), "toload.apk");
File dexOutputDir = context.getDir("dex", 0);
DexClassLoader dexClass = new DexClassLoader(file.getPath(),
        dexOutputDir.getPath(), null, ClassLoader.getSystemClassLoader());
// get class by name
Class<?> loadClass = dexClass.loadClass("com.example.testload.MainActivity");
// create an instance of the class
Constructor<?> loadConstructor = loadClass.getConstructor(new Class[] {});
Object instance = loadConstructor.newInstance(new Object[] {});
// use reflection to invoke method
Method onLoadFun = loadClass.getMethod("loadedLogTest");
onLoadFun.invoke(instance);
```

Content

- Background
- aDFAer - Android Data Flow Analyzer
- Janus - Detect Reflection
- Experiment
- Future Work

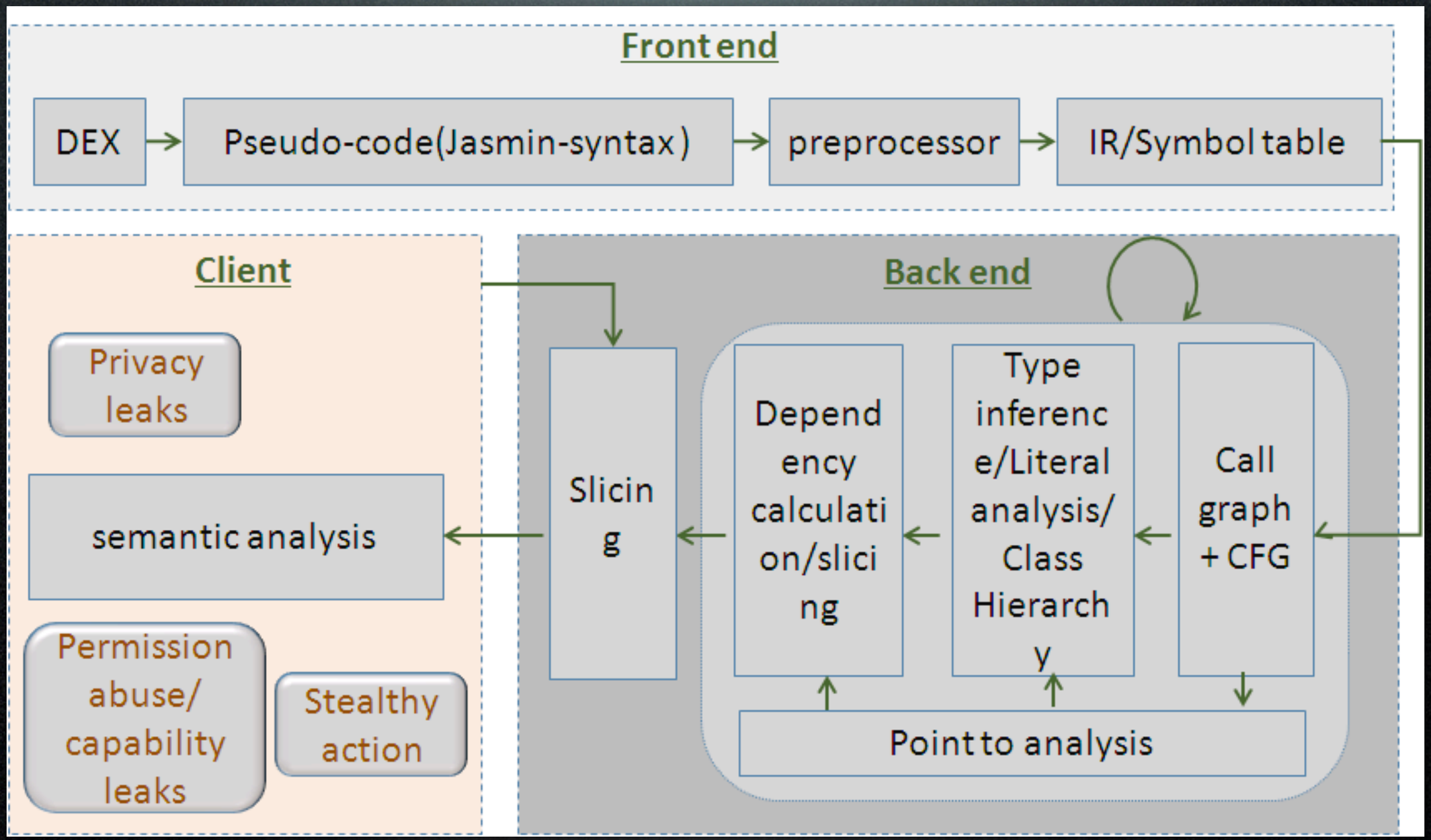
aDFAer Overview



aDFAer Overview

- Designed to solve different problems (privacy leaks, capability leaks, etc.)
  - Client & Engine
- aDFAer is basically a data flow analysis engine
  - It dose not care what the data means, but to track the data (defined by client) transferred in the app
- Developing client on aDFAer engine to solve different problems
  - Obey the rule “sources->sanitizers ->sinks” to write policy. Then it will give what you want

aDFAer Architecture



Test Cases

- There are some examples show the ability of aDFAer
  - Reaching definition
  - Path merging
  - Inter-procedural
  - Dealing with Obfuscator
  - Reachable analysis

01-Reaching Definition

- MainActivity.java

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    String SMS_URI_ALL = "content://sms/";
    String SMS_URI_INBOX = "content://sms/inbox";
    String SMS_URI_SEND = "content://sms/sent";
    String SMS_URI_DRAFT = "content://sms/draft";

    Uri uri = Uri.parse(SMS_URI_SEND);

    String[] projection = new String[] { "_id", "address", "person",
            "body", "date", "type" };
    Cursor cusor = managedQuery(uri, projection, null, null,
            "date desc");

}
```

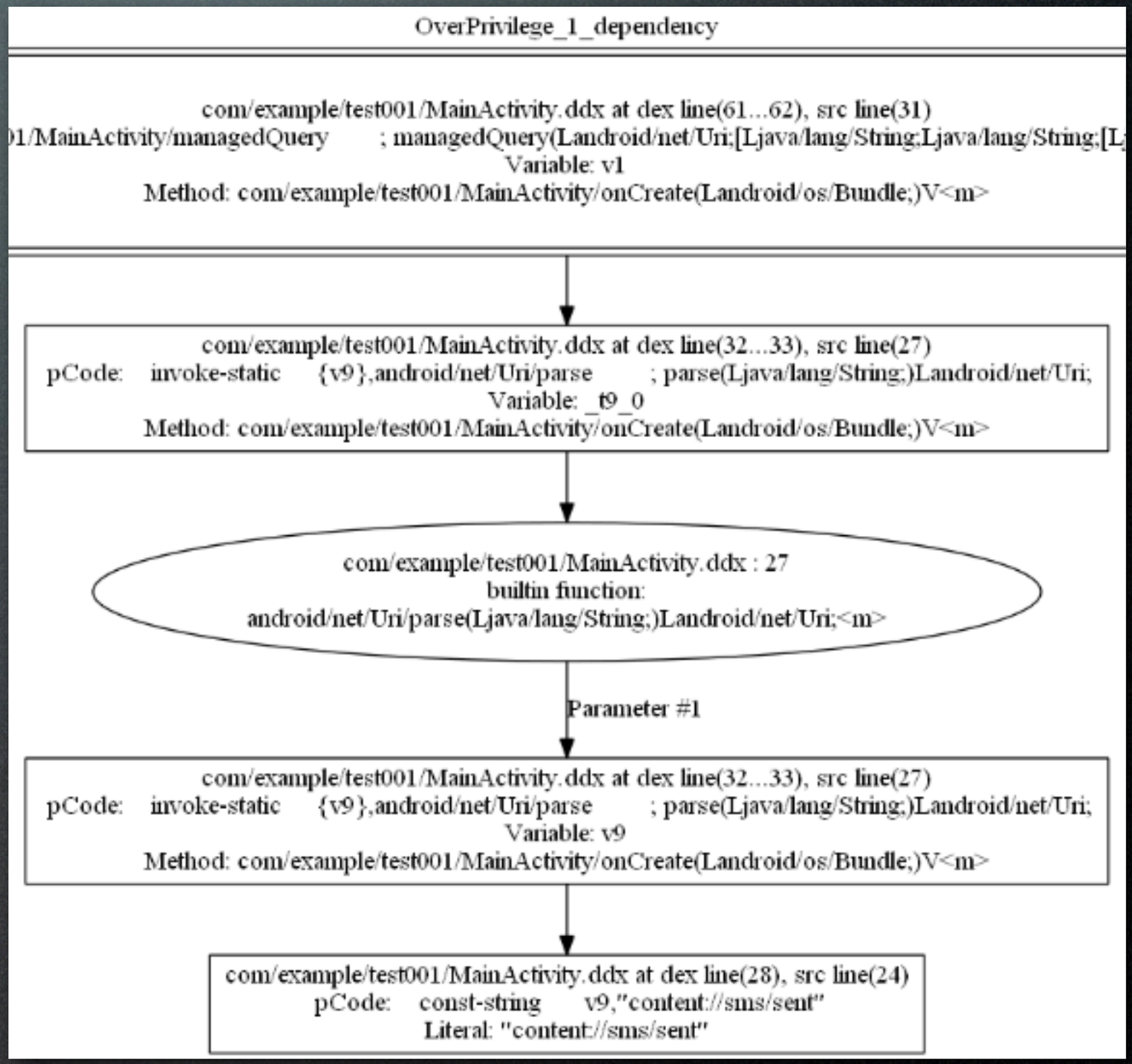
- aDFAer output

```
Total Graph Count: 1
Total Privilege Used Count: 1
---------------------------------
["content://sms/sent"]


*********************************
Over Privilege Analysis END
*********************************
```

O1-Reaching Definition

- aDFAer generate dependency graph



02-Path Merging

- MainActivity.java

```
int a = 10, b = 5;

String SMS_URI_ALL = "content://sms/";
String SMS_URI_INBOX = "content://sms/inbox";
String SMS_URI_SEND = "content://sms/sent";
String SMS_URI_DRAFT = "content://sms/draft";

Uri uri = null;
if(a > b)
    uri = Uri.parse(SMS_URI_SEND);
else
    uri = Uri.parse(SMS_URI_DRAFT);

String[] projection = new String[] { "_id", "address", "person",
        "body", "date", "type" };
Cursor cusor = managedQuery(uri, projection, null, null,
        "date desc");
```

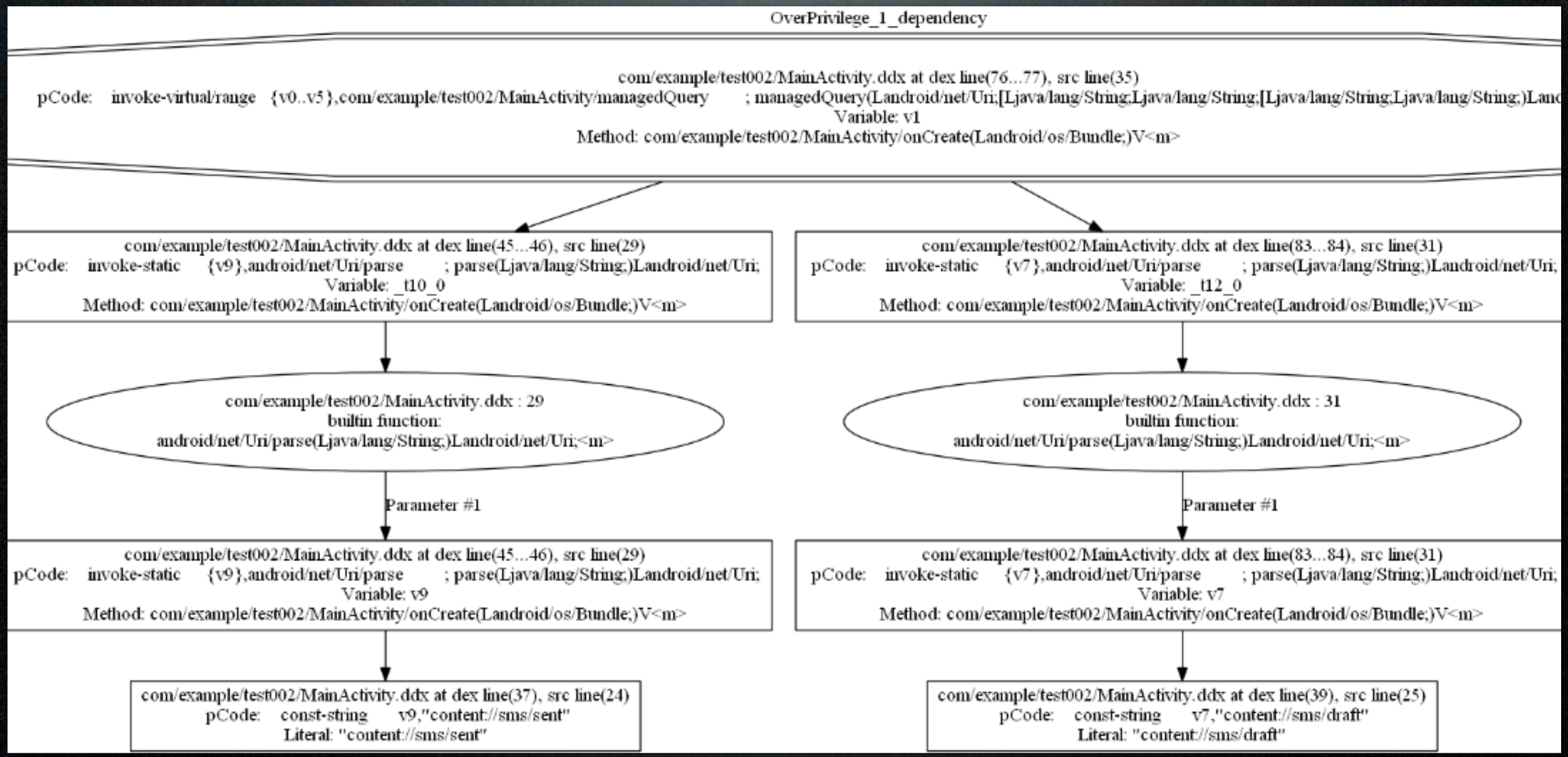
- aDFAer output

```
Total Graph Count: 1
Total Privilege Used Count: 2
---------------------------------
["content://sms/draft", "content://sms/sent"]


*********************************
Over Privilege Analysis END
*********************************
```

02-Path Merging

- aDFAer generate dependency graph



03-Inter-procedural

- MainActivity.java

```
        String[] projection = new String[] { "_id", "address", "person",
                "body", "date", "type" };
        inter myInter = new inter();

        Cursor cusor = managedQuery(myInter.interUri(), projection, null, null,
                "date desc");
```

- Inter.java

```
public class inter {
    String SMS_URI_ALL = "content://sms/";
    String SMS_URI_INBOX = "content://sms/inbox";
    String SMS_URI_SEND = "content://sms/sent";
    String SMS_URI_DRAFT = "content://sms/draft";

    public Uri interUri()
    {
        int a = 10, b = 5;

        Uri retMe = null;

        if(a > b)
            retMe = Uri.parse(SMS_URI_SEND);
        else
            retMe = Uri.parse(SMS_URI_DRAFT);

        return retMe;
    }
}
```

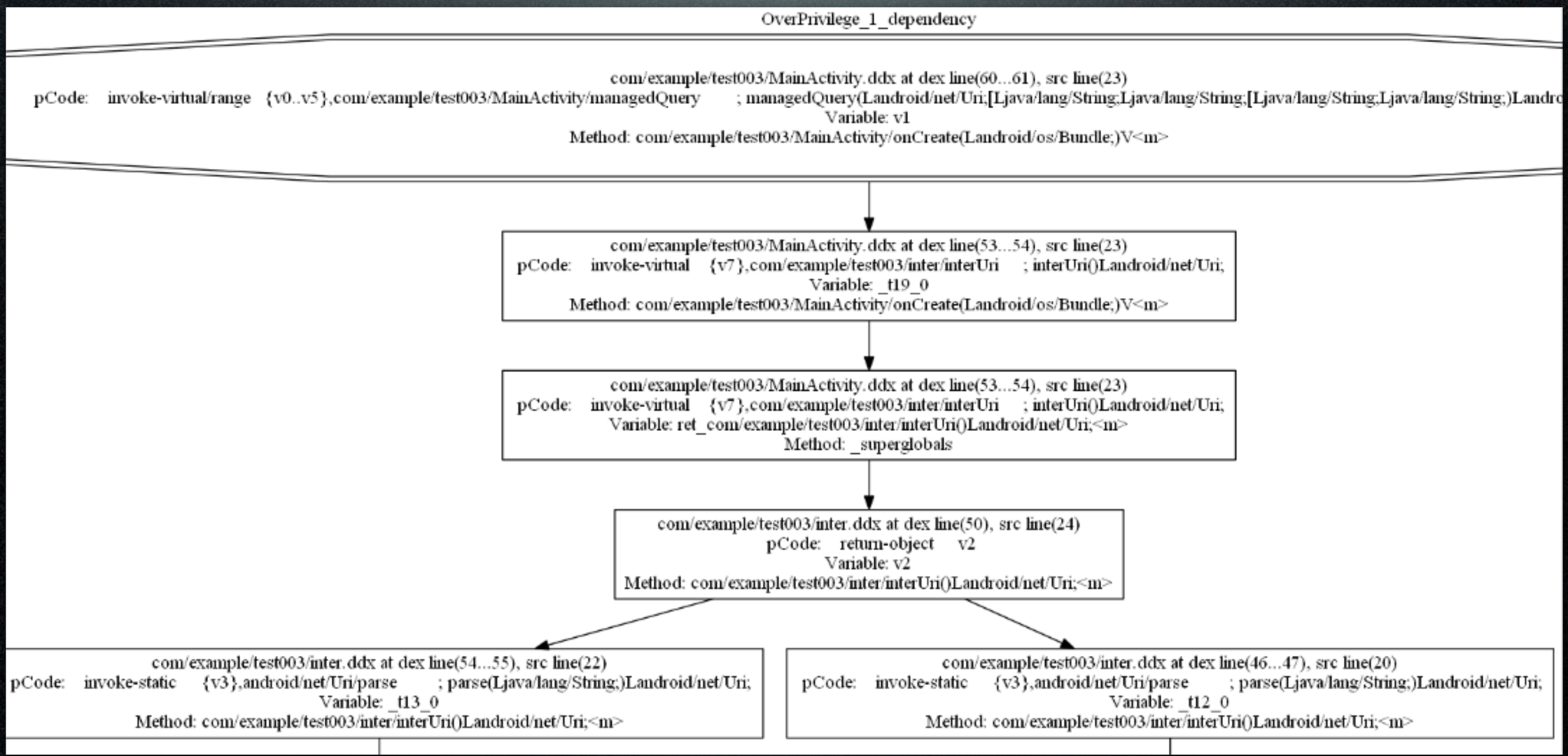
- aDFAer output

```
Total Graph Count: 1
Total Privilege Used Count: 3
---------------------------------
[_system.return_com/example/test003/inter/interUri()Landroid/net/Uri;<method>, "content://sms/draft", "content://sms/sent"]


*********************************
Over Privilege Analysis END
*********************************
```

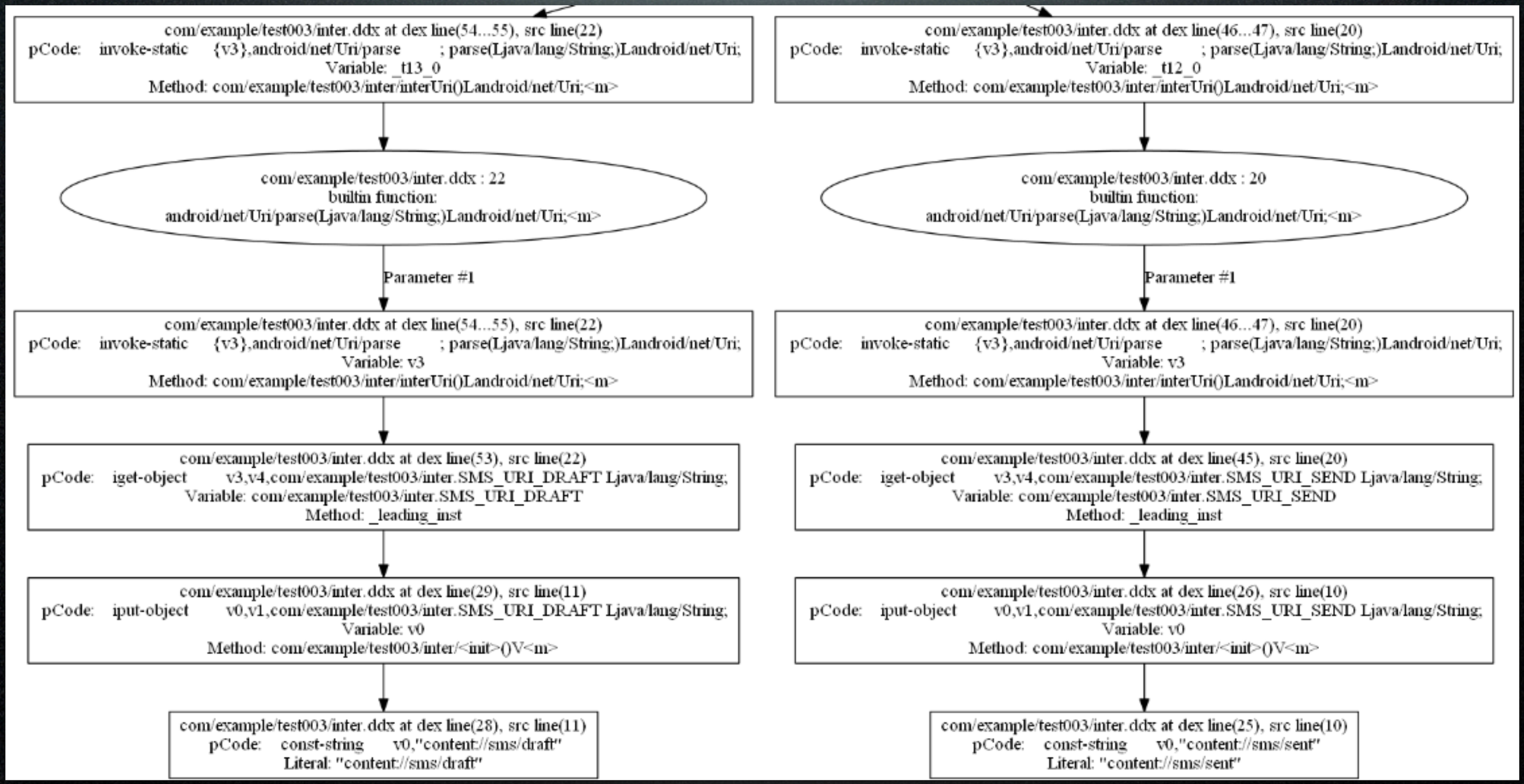
03-Inter-procedural

- aDFAer generate dependency graph



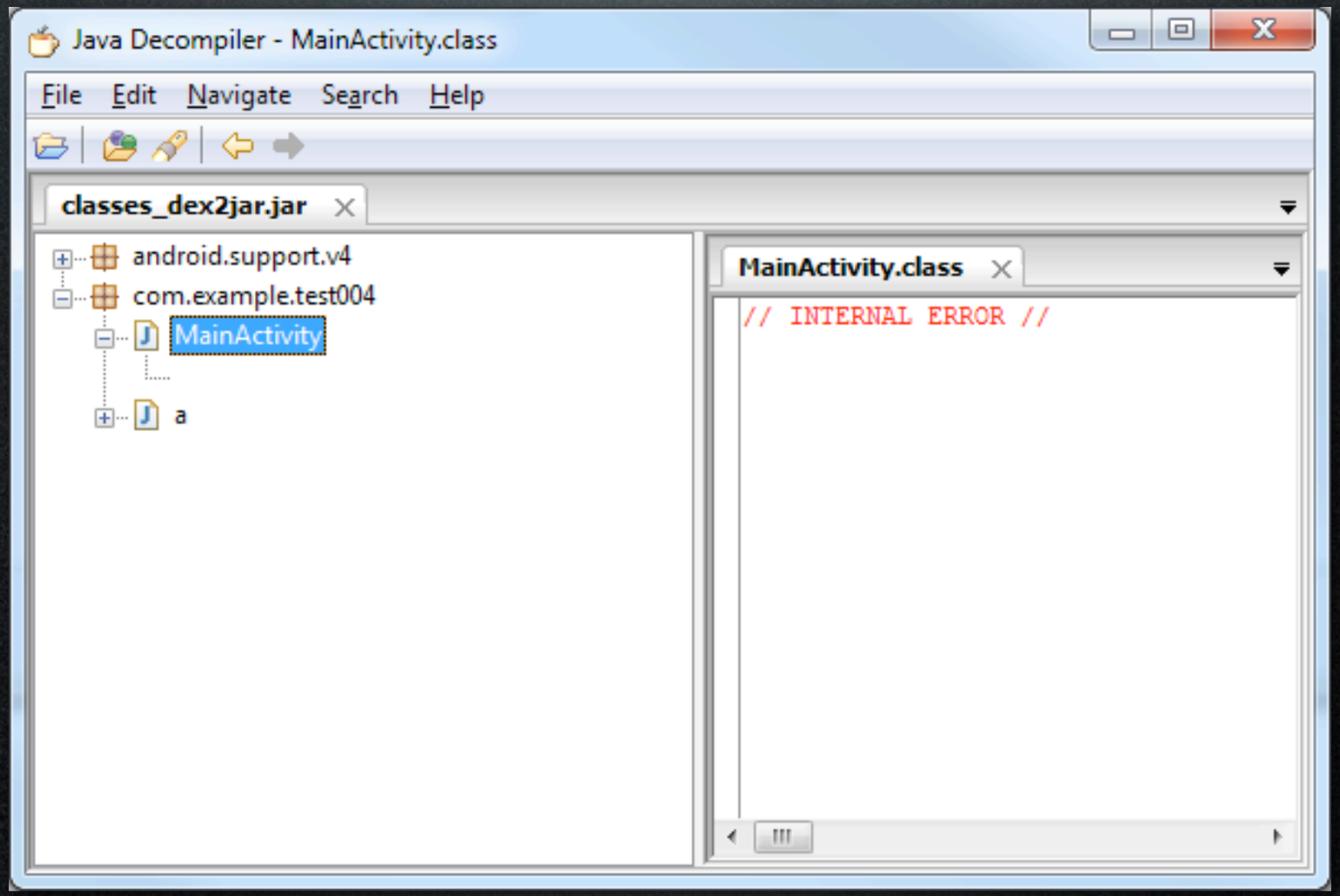
03-Inter-procedural

- aDFAer generate dependency graph cont.



04-Optimizer and Obfuscator

- Using ProGuard to protect test case 03
- Decomplier could not run properly



04-Optimizer and Obfuscator

- aDFAer output

```
Total Graph Count: 1
Total Privilege Used Count: 3
----------------------------------
[_system.return_com/example/test004/a/a(II)Landroid/net/Uri;<method>, "content://sms/draft", "content://sms/sent"]


**********************************
Over Privilege Analysis END
**********************************
```

- Obfuscator is only an obstacle for human reading, but not for automatic analysis tool

05-Reachable Analysis

- MainActivity.java

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);


}

public void unreachable()
{
    String SMS_URI_ALL = "content://sms/";
    String SMS_URI_INBOX = "content://sms/inbox";
    String SMS_URI_SEND = "content://sms/sent";
    String SMS_URI_DRAFT = "content://sms/draft";

    Uri uri = Uri.parse(SMS_URI_SEND);

    String[] projection = new String[] { "_id", "address", "person",
            "body", "date", "type" };
    Cursor cusor = managedQuery(uri, projection, null, null,
            "date desc");

}
```

- aDFAer output

```
not reachable!!!
Total Graph Count: 0
Total Privilege Used Count: 0
---------------------------------
[]


*********************************
Over Privilege Analysis END
*********************************
```

Writing Clients

- To solve different problem, you need model the problem that you are concentrating
- Malicious operation grammar

⟨*stat-item*⟩ ::= ‘START’ ⟨*statement*⟩ ‘:’ ⟨*risklevel*⟩ ‘:’ ⟨*stat-description*⟩ ‘END’

⟨*statement*⟩ ::= ⟨*method-stat*⟩ ‘{’ ⟨*rules*⟩ ‘}’

⟨*rules*⟩ ::= ϵ | (⟨*rules*⟩ ‘{’ ⟨*rule*⟩ ‘}’)

⟨*rule*⟩ ::= ⟨*arg-index*⟩ ‘:’ (‘l:’ ⟨*query-statements*⟩ | ‘t:’ ⟨*query-statements*⟩ | ‘v:’ ⟨*query-statements*⟩ | ‘m:’ ⟨*statement*⟩)

⟨*query-statements*⟩ ::= ⟨*query-statements*⟩ ‘,’ ⟨*query-statement*⟩

⟨*query-statement*⟩ ::= ⟨*regular expression*⟩

⟨*arg-index*⟩ ::= ⟨*unsigned integer*⟩

⟨*risklevel*⟩ ::= ‘0’ | ‘1’ | ‘2’

Fig. 1. Sources/malicious operation grammar of aDFAer

Writing Clients

- Sinks grammar

⟨*stat-item*⟩ ::= ‘START’ ⟨*statement*⟩ ‘:’ ⟨*risklevel*⟩ ‘:’ ⟨*stat-description*⟩ ‘END’

⟨*statement*⟩ ::= ⟨*method-stat*⟩ ‘{’ ⟨*sources*⟩ ⟨*rules*⟩ ‘}’

⟨*sources*⟩ ::= ⟨*sources*⟩ ‘{’ ⟨*source*⟩ ‘}’

⟨*source*⟩ ::= ⟨*arg-index*⟩

⟨*rules*⟩ ::= ϵ | (⟨*rules*⟩ ‘{’ ⟨*rule*⟩ ‘}’)

⟨*rule*⟩ ::= ⟨*arg-index*⟩ ‘:’ (‘l:’ ⟨*query-statements*⟩ | ‘t:’ ⟨*query-statements*⟩ | ‘v:’ ⟨*query-statements*⟩ | ‘m:’ ⟨*statement*⟩)

⟨*query-statements*⟩ ::= ⟨*query-statements*⟩ ‘,’ ⟨*query-statement*⟩

⟨*query-statement*⟩ ::= ⟨*regular expression*⟩

⟨*arg-index*⟩ ::= ⟨*unsigned integer*⟩

⟨*risklevel*⟩ ::= ‘0’ | ‘1’ | ‘2’

Fig. 2. Sinks grammar of aDFAer

Clients Demo

- Get local phone number

```
<?xml version="1.0" encoding="UTF-8"?>
- <root xsi:noNamespaceSchemaLocation="../aDFAerPolicy.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    - <start>
        <hit> android/telephony/TelephonyManager/getLine1Number() </hit>
      </start>
    - <start>
        <hit> android/accounts/AccountManager/getAccounts() </hit>
      </start>
    - <start>
        <hit> android/telephony/TelephonyManager/getDeviceId() </hit>
      </start>
```

Clients Demo

- Dynamic code loading

```
- <start>
    - <hit>
            <!-- invoke-virtual {v0,v5},dalvik/system/DexClassLoader/loadClass ; loadClass
            (Ljava/lang/String;)Ljava/lang/Class; -->
        dalvik/system/DexClassLoader/loadClass(Ljava/lang/String;)
    </hit>
    <confirm type="t" serial="0"> .*dalvik/system/DexClassLoader.* </confirm>
    <risklevel> 1 </risklevel>
    <description> Dynamic loading class </description>
</start>
```

```
Output:
Find malicious operation 1 at:
        - dex file name: com/plankton/device/android/service/g.ddx
        - dex file offset: 75...76
        - src file offset: -1
        - description: Dynamic loading class
        - risk level: MEDIUM

Total confirmed malicious operations: 1
```

Clients Demo

- Query contacts

```
- <start>
      <hit> _FRAMEWORK/managedQuery(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;
          [Ljava/lang/String;Ljava/lang/String;) </hit>
      <confirm type="t" serial="1"> .*android/net/Uri.* </confirm>
      <confirm type="v" serial="1"> .*_FRAMEWORK.android/provider/Contacts.* </confirm>
      <risklevel> 1 </risklevel>
      <description> query contacts </description>
  </start>
```

```
Find malicious operation 3 at:
        - dex file name: ll/ap/ken/LlApKenActivity.ddx
        - dex file offset: 164...165
        - src file offset: 90
        - description: query contacts
        - risk level: MEDIUM
Output:
Find malicious operation 4 at:
        - dex file name: ll/ap/ken/LlApKenActivity.ddx
        - dex file offset: 201...202
        - src file offset: 107
        - description: query contacts
        - risk level: MEDIUM

Total confirmed malicious operations: 4
```

Clients Demo

- Privacy leaks
  - Craig's POC
    - http://secur3.us/DC21Slides.pdf
- aDFAer output
  - http://program-analysis.oicp.net:8080/co-site/Janus/Reflection/syscan2013/t1/res.svg

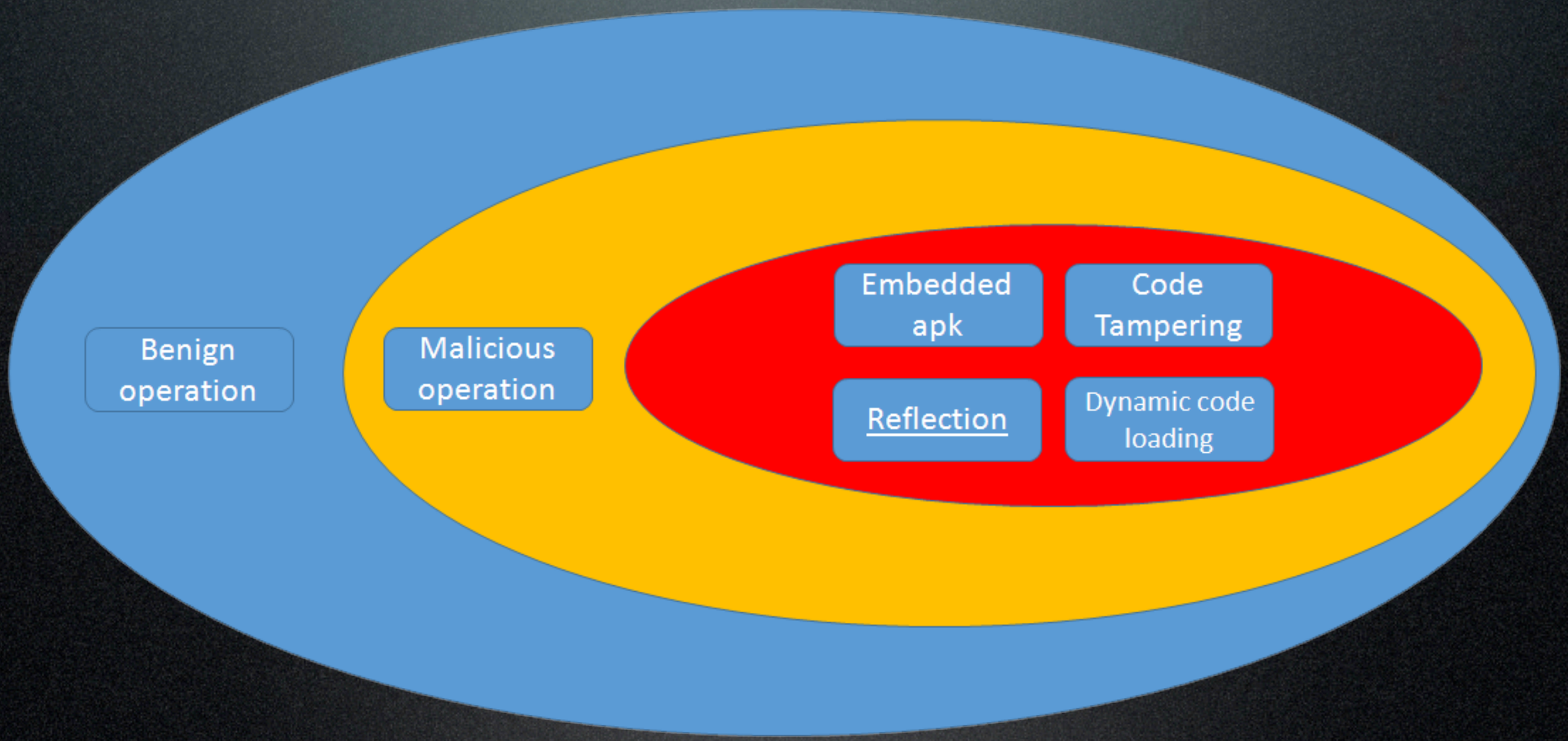
Content

- Background
- aDFAer - Android Data Flow Analyzer
- Janus - Detect Reflection
- Experiment
- Future Work

Janus

- Sub project of aDFAer, the project tries to develop an auxiliary tool for finding advanced malware
- We only discuss how to detect malware using reflection in this topic

Risk Level


Benign operation
Malicious operation
Embedded apk
Code Tampering
Reflection
Dynamic code loading

Coexisting Detect Work

- Static Analysis of Dalvik Bytecode and Reflection in Android
  - http://projekter.aau.dk/projekter/files/63640573/rapport.pdf
- V. Benjamin Livshits, John Whaley, Monica S. Lam: Reflection Analysis for Java. APLAS 2005:139-160
  - http://www.dblp.org/db/conf/aplas/aplas2005.html

Art of Using Reflection

Technique	Difficulty	Example
Obj receiver and method name come from configure file, network etc.	hard	1. String className = r.readLine(); 2. Class c = Class.forName(className); 3. Object o = c.newInstance(); 4. T t = (T) o;
Obj receiver or method name is obscured	medium	1. String className = decode("xyz"); 2. Class c = Class.forName(className); 3. Object o = c.newInstance(); 4. T t = (T) o;
Basic	easy	1. String className = "java.lang.String"; 2. Class c = Class.forName(className); 3. Object o = c.newInstance(); 4. T t = (T) o;

Case Study

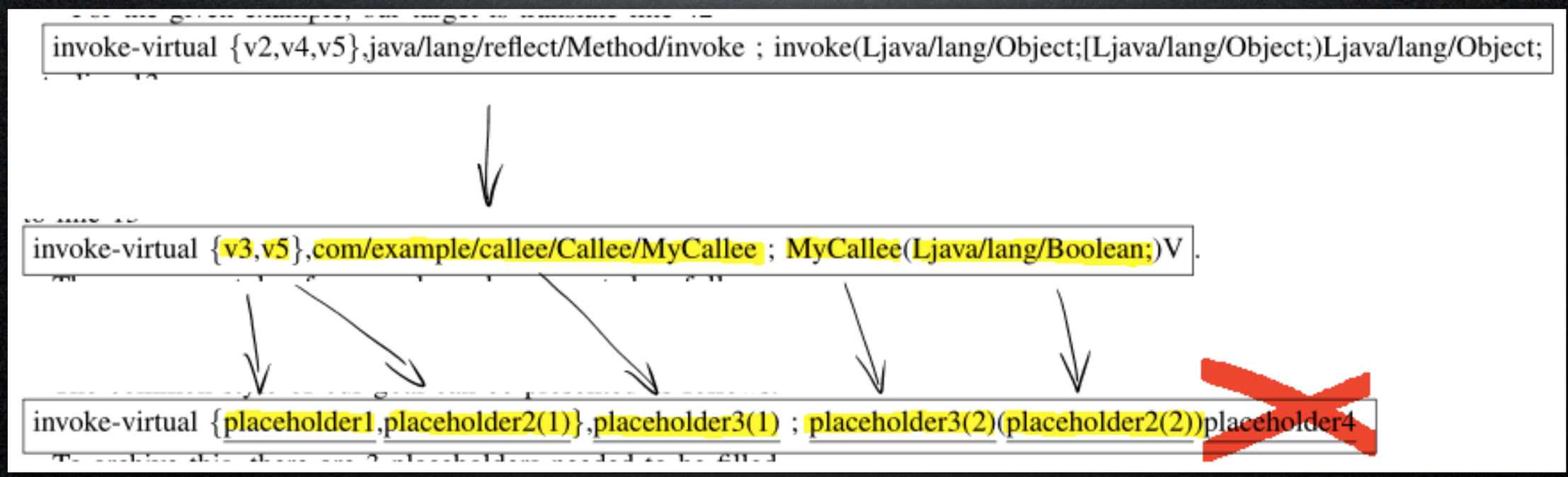
```
Callee myCallee = new Callee();
myCallee.MyCallee(true);

try {
        Class c = Class.forName("com.example.callee.Callee");
        Object o = c.newInstance();
        Method method = c.getMethod("MyCallee", new Class[]{Boolean.class});
        method.invoke(o, false);
    ...
```

```
.line 23
        const-string    v5, "com.example.callee.Callee"
        invoke-static   {v5}, Ljava/lang/Class;->forName    ; forName(Ljava/lang/String;)Ljava/lang/Class;
        move-result-object      v0
:12b632:
.line 24
        invoke-virtual  {v0}, Ljava/lang/Class;->newInstance        ; newInstance()Ljava/lang/Object;
        move-result-object      v4
:12b63a:
.line 25
        const-string    v5, "MyCallee"
        const/4 v6, 1
        new-array       v6, v6, [Ljava/lang/Class;
        const/4 v7, 0
        const-class     v8, Ljava/lang/Boolean;
        aput-object     v8, v6, v7
        invoke-virtual  {v0, v5, v6}, Ljava/lang/Class;->getMethod    ; getMethod(Ljava/lang/String;[Ljava/lang/Class;)Ljava/lang/reflect/Method;
        move-result-object      v2
:12b656:
.line 26
        const/4 v5, 1
        new-array       v5, v5, [Ljava/lang/Object;
        const/4 v6, 0
        const/4 v7, 0
        invoke-static   {v7}, Ljava/lang/Boolean;->valueOf  ; valueOf(Z)Ljava/lang/Boolean;
        move-result-object      v7
        aput-object     v7, v5, v6
        invoke-virtual  {v2, v4, v5}, Ljava/lang/reflect/Method;->invoke        ; invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/Object;
```

How to Detect

- Placeholder1 - “this” argument passed to the callee
- Placeholder2 - arguments passed to the callee
- Placeholder3 - class name and method name
- Placeholder4 - useless return value



How to Detect

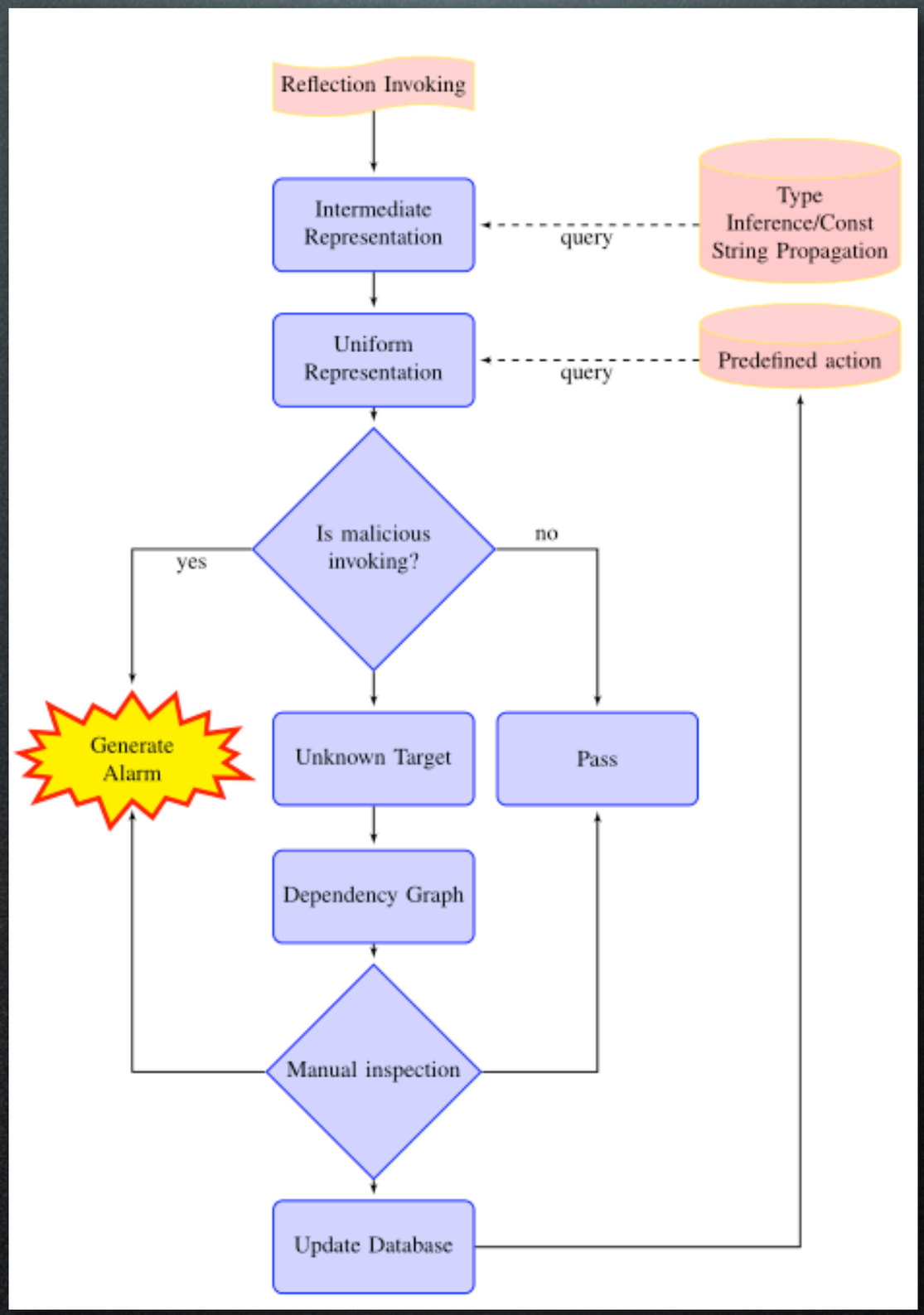
- Method for getting class object

Method for getting Class Object	Examples
getClass()	String str = ”abc”; Class c1 = str.getClass();
Class.getSuperclass()	Button b = new Button(); Class c1 = b.getClass(); Class c2 = c1.getSuperclass();
static method Class.forName()	Class c1 = Class.forName (”java.lang.String”); Class c2 = Class.forName (”java.awt.Button”); Class c3 = Class.forName(”java.util.LinkedList$Entry”); Class c4 = Class.forName (”I”); Class c5 = Class.forName (”[I”);
.class grammar	Class c1 = String.class; Class c2 = java.awt.Button.class; Class c3 = Main.InnerClass.class; Class c4 = int.class; Class c5 = int[].class;
Type grammar of primitive wrapper classes	Class c1 = Boolean.TYPE; Class c2 = Byte.TYPE; Class c3 = Character.TYPE; Class c4 = Short.TYPE; Class c5 = Integer.TYPE; Class c6 = Long.TYPE; Class c7 = Float.TYPE; Class c8 = Double.TYPE; Class c9 = Void.TYPE;

Algorithm

Input:
1: Literal analysis result: literAnalysis
2: Type analysis result: typeAnalysis
3: Dependency analysis result: depAnalysis
4: Built-in reflection call: RC
Output:
5: Uniform representation for Reflection call
6:
7: **procedure** DFS
8: **for** each CfgNode $\in$ depAnalysis.getAllNode() **do**
9: **if** CfgNode $\equiv$ RC **then**
10: Reflection2Uniform(CfgNode)
11: **end if**
12: **end for**
13: **end procedure**
14:
15: **procedure** REFLECTION2UNIFORM(CfgNode)
16: placeholder1 $\leftarrow$ CfgNode.getParam(2)
17: **for** each ElementVariable $\in$ CfgNode.getParam(3).getArrayElements() **do**
18: placeholder2(1) $\leftarrow$ ElementVariable
19: **end for**
20: CfgNodeX $\leftarrow$ depAnalysis.getDef(CfgNode.getParams(1), CfgNode) ▷ Goto "getMethod" or "getDeclaredMethod"
21: placeholder3(1) $\leftarrow$ typeAnalysis.query(CfgNodeX.getParam(1), CfgNodeX) $\circ$ literalAnalysis.query(CfgNodeX.getParam(2), CfgNodeX)
22: placeholder3(2) $\leftarrow$ literalAnalysis.query(CfgNodeX.getParam(2), CfgNodeX)
23: **for** each ElementVariable $\in$ CfgNodeX.getParam(3).getArrayElements() **do**
24: placeholder2(2) $\leftarrow$ typeAnalysis.query(ElementVariable, CfgNodeX) ▷ not be applied in the first iteration, for it may be assigned to null
25: **end for**
26: **end procedure**

Workflow

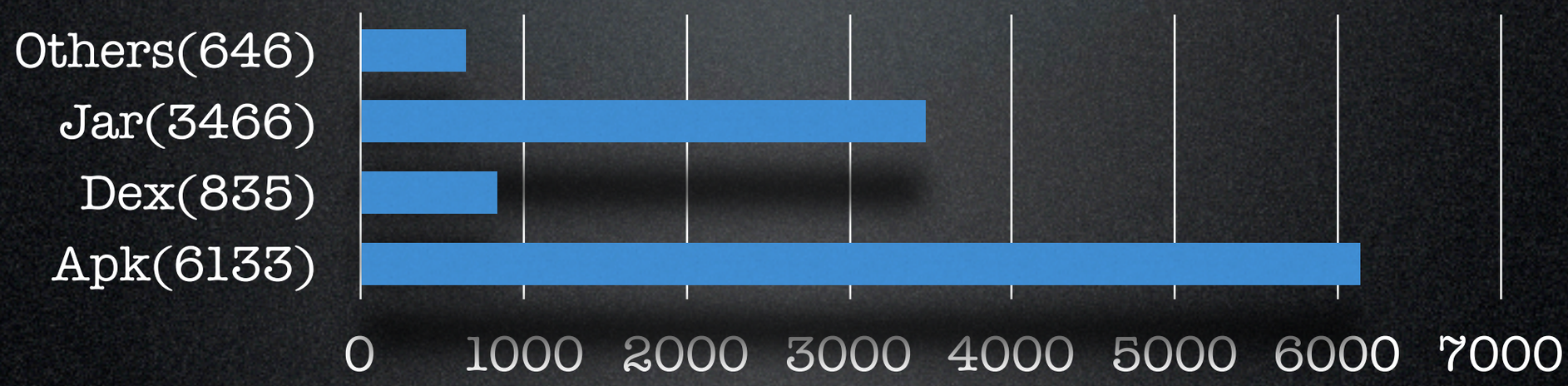

Reflection Invoking
Intermediate Representation
query
Type Inference/Const String Propagation
Uniform Representation
query
Predefined action
Is malicious invoking?
yes
no
Generate Alarm
Unknown Target
Pass
Dependency Graph
Manual inspection
Update Database

Content

- Background
- aDFAer - Android Data Flow Analyzer
- Janus - Detect Reflection
- Experiment
- Future Work

Test Samples

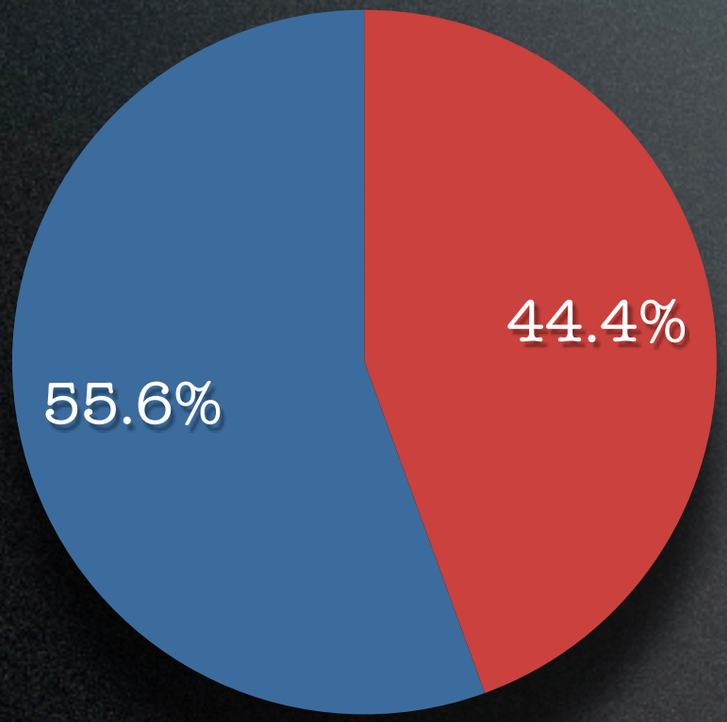
- mumayi.com - **4597** Apps
- VirusShare_Android_20130506.zip
  - Total 11080 samples from VirusShare
  - Others include ELF, PE, etc. format file
- We test all APK samples - **6133** Apps



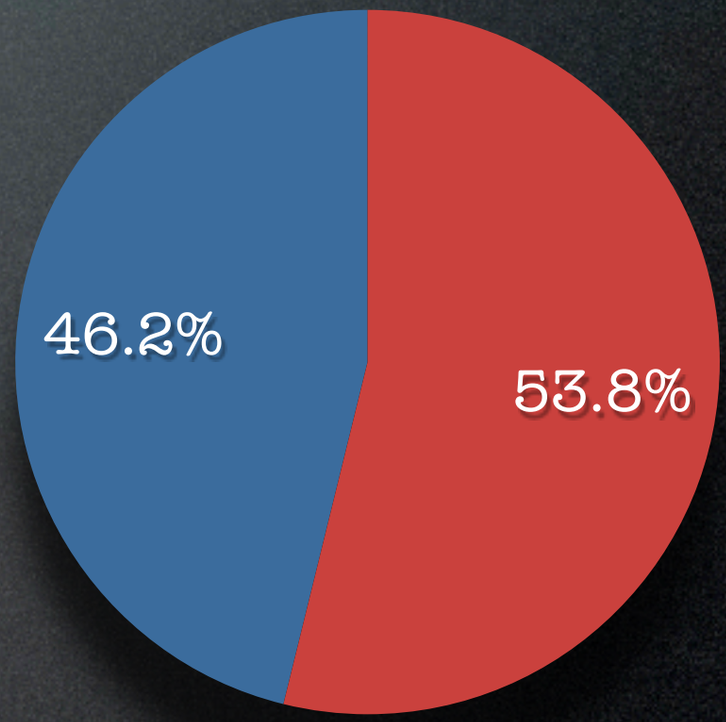
Percentage of Reflection

- We can easily confirm reflection by statically scanning for "java/lang/reflect/Method/invoke" in smali code

VirusShare (6133)



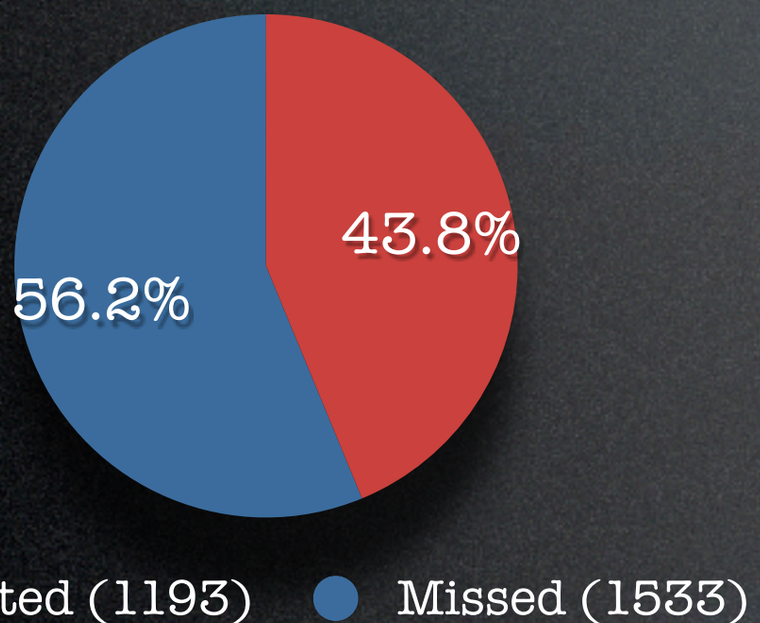
mumayi.com (4597)



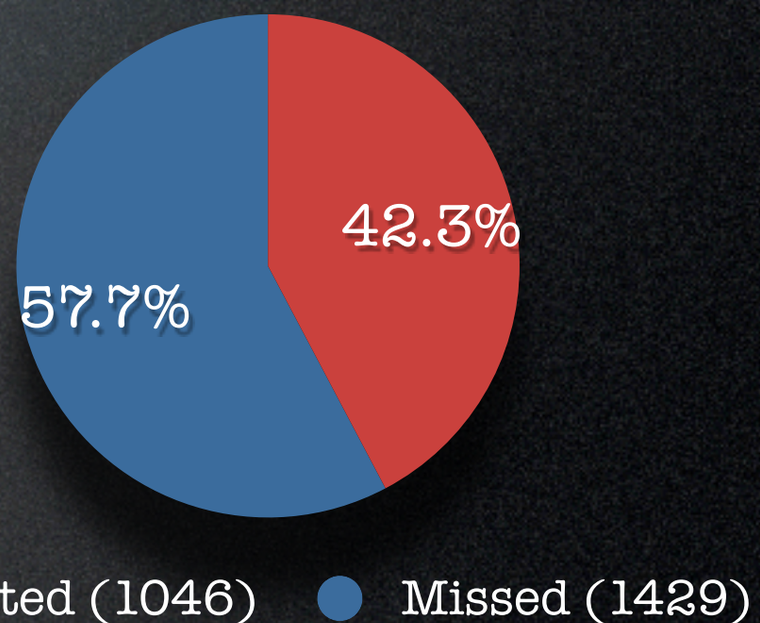
Detected Reflection by Janus

- Detected means - confirm class/method name and arguments
- Improve to reach 100%
  - Used in Support Library
  - A full-fledged call graph complement module is needed
  - ...

VirusShare (2726)



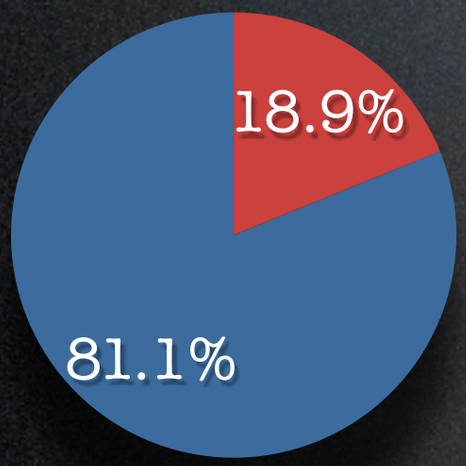
mumayi.com (2475)



Identified Malicious Operation

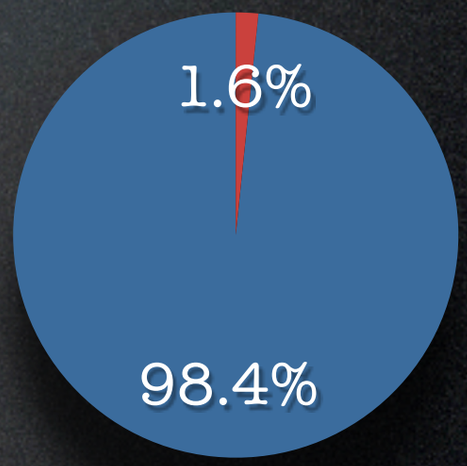
- We set only 4 rules
  - .*setMobileDataEnabled.* = -1:Operating your 3G module
  - .*getLine1Number.* = -1:Query your local phone number
  - .*SmsManager.* = -1:Operating your SMS message
  - .*enableDataConnectivity.* = -1:Operating your data connectivity

VirusShare (1193)



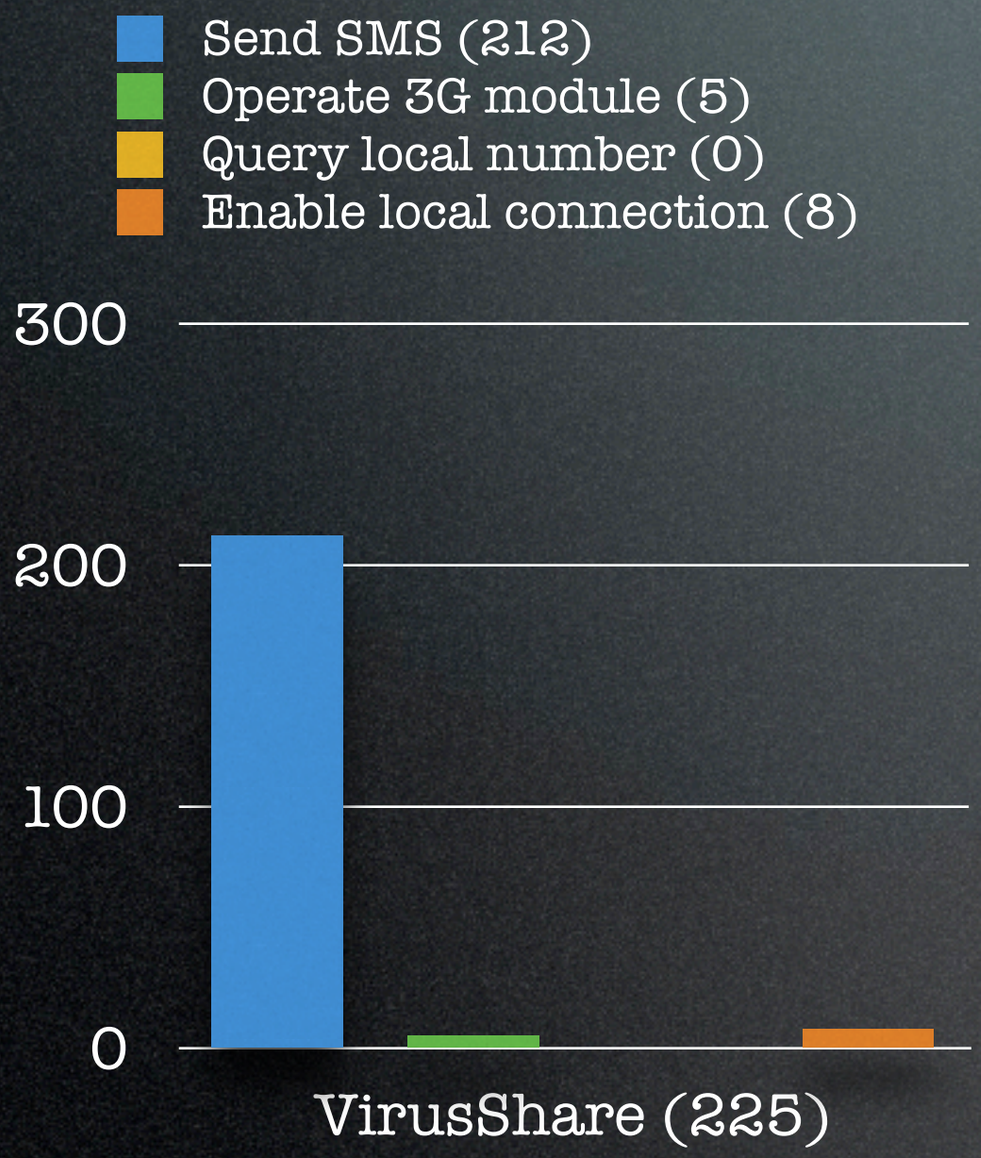
Contains malicious operation (225)
Unknown (968)

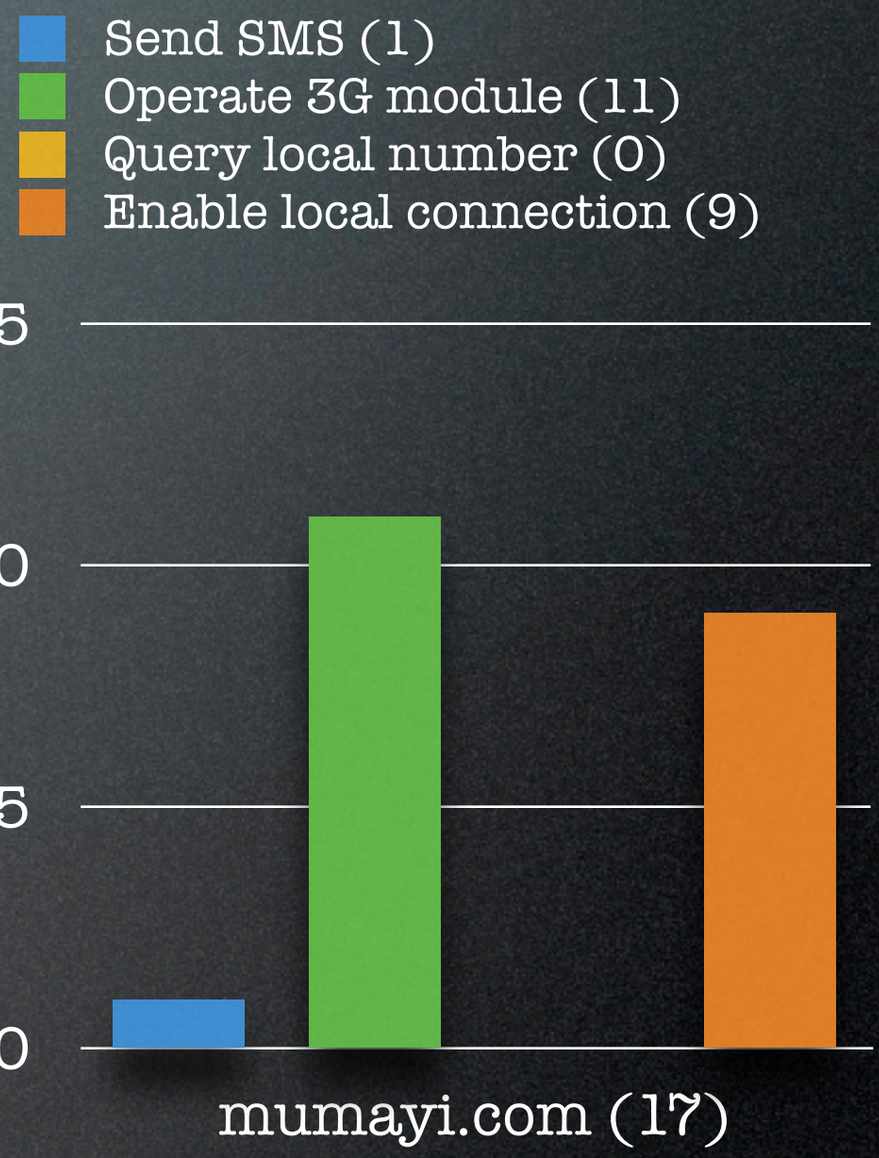
mumayi.com (1046)



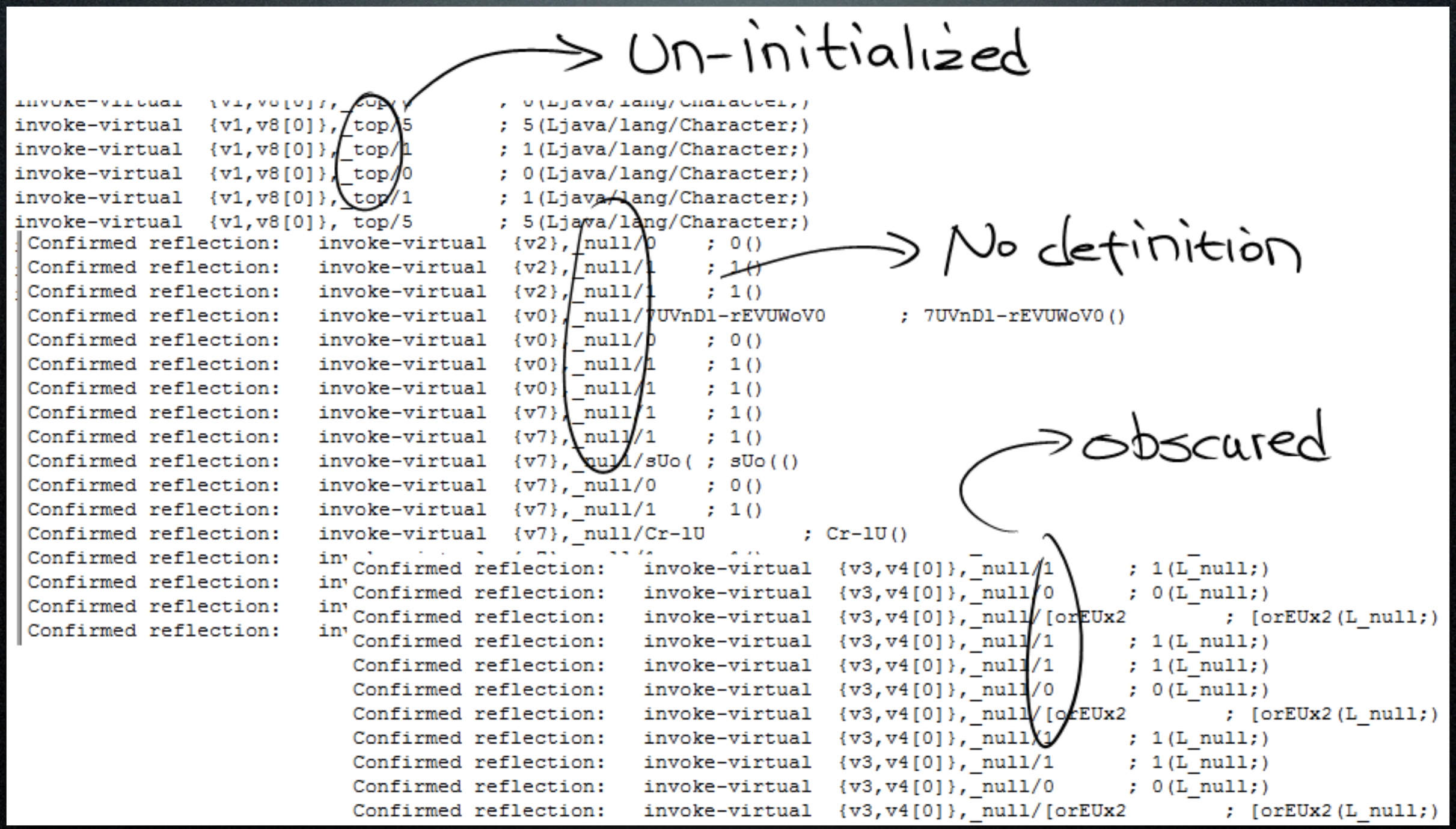
Contains malicious operation (17)
Unknown (1029)

Identified Malicious Operation





Manual Inspection



Manual Inspection

- mumayi.com
  - http://program-analysis.oicp.net:8080/co-site/Janus/Reflection/syscan2013/manual/mumayi.out.log
- VirusShare
  - http://program-analysis.oicp.net:8080/co-site/Janus/Reflection/syscan2013/manual/virusshare.out.log

Case 01

- obad

```
**********
Reflection Analysis BEGIN(Outer)
**********

Total sinks: 3

Checking sink at:
	- dex line: 1565...1566
	- src line: -1
	- pseudo code:	invoke-virtual {v0,v2,v1},java/lang/reflect/Method/invoke ; invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/Object;
	- enclosing file name: com/android/system/admin/lOClOOI.ddx
Confirmed reflection:	invoke-virtual {v2,v1[0]},_null/0 ; 0(Ljava/lang/String;)
Confirmed reflection:	invoke-virtual {v2,v1[0]},_null/65506 ; 65506(Ljava/lang/String;)
Confirmed reflection:	invoke-virtual {v2,v1[0]},_null/41 ; 41(Ljava/lang/String;)

Checking sink at:
	- dex line: 1598
	- src line: -1
	- pseudo code:	invoke-virtual {v0,v6,v1},java/lang/reflect/Method/invoke ; invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/Object;
	- enclosing file name: com/android/system/admin/lOClOOI.ddx
Confirmed reflection:	invoke-virtual {v6,v1[0]},_null/41 ; 41(L[B;)
Confirmed reflection:	invoke-virtual {v6,v1[0]},_null/65501 ; 65501(L[B;)
Confirmed reflection:	invoke-virtual {v6,v1[0]},_null/0 ; 0(L[B;)
```

Case 02

- shoujijianting_V1.2.0_mumayi_0fcd2.apk


```
Confirmed reflection:   invoke-virtual  {v4,v5[0]},android/net/NetworkInfo$State/setMobileDataEnabled   ; setMobileDataEnabled
(Ljava/lang/Boolean;)
Output: - find reflection, Description:          - Operating your 3G module
Confirmed reflection:   invoke-virtual  {v4,v5[0]},android/net/NetworkInfo$State/setMobileDataEnabled   ; setMobileDataEnabled
(Ljava/lang/Boolean;)
Output: - find reflection, Description:          - Operating your 3G module
Confirmed reflection:   invoke-virtual  {v4,v5[0]},android/net/NetworkInfo$State/setMobileDataEnabled   ; setMobileDataEnabled
(Ljava/lang/Boolean;)
```

Case 02

腾讯手机管家
手机安全管理先锋

首页　下载

<< 返回安全实验室

检测报告

软件名称：Listener
操作系统：android
开发方名称：
风险级别：

高　中　低

软件包信息

软件名称：Listener

软件大小：960 kb

操作平台：Android平台

检查结果

安全级别

危险　系统冲突　谨慎　安全　未知

推荐操作

可放心安装!

立即下载网秦安全，时刻保护您的手机安全。　查看详情 >>

virustotal

SHA256:　a93af4a825a79c1a270cee99567e7

File name:　shoujijianting_V1.2.0_mumayi_0fc

Detection ratio:　0 / 46

Analysis date:　2013-08-21 03:04:15 UTC (22 hours, 34 minutes ago)

Case 02

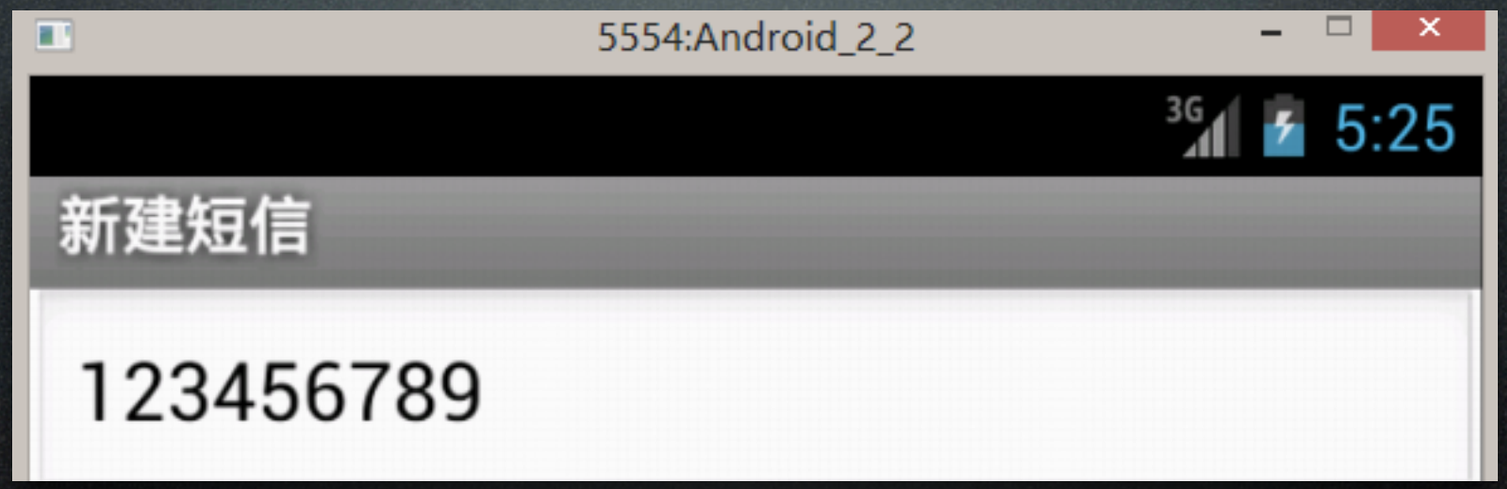
权限列表

• 高危 • 危险 • 普通

录音（使用AudioRecord）	已使用	android.permission.RECORD_AUDIO
读取联系人信息	已使用	android.permission.READ_CONTACTS
监视、修改有关拨出电话	已使用	android.permission.PROCESS_OUTGOING_CALLS
监控接收短信	已使用	android.permission.RECEIVE_SMS
写短信	未使用	android.permission.WRITE_SMS
读取短信	未使用	android.permission.READ_SMS
读取wifi网络状态	已使用	android.permission.ACCESS_WIFI_STATE
改变WIFI连接状态	已使用	android.permission.CHANGE_WIFI_STATE
读取系统日志	已使用	android.permission.READ_LOGS
挂载、反挂载外部文件系统	未使用	android.permission.MOUNT_UNMOUNT_FILESYSTEMS
接收开机启动广播	已使用	android.permission.RECEIVE_BOOT_COMPLETED
获取粗略的位置（通过wifi...	已使用	android.permission.ACCESS_COARSE_LOCATION
获取精确的位置（通过GPS）	已使用	android.permission.ACCESS_FINE_LOCATION
读取网络状态（2G或3G）	已使用	android.permission.ACCESS_NETWORK_STATE
写外部存储器（如：SD卡）	未使用	android.permission.WRITE_EXTERNAL_STORAGE
读取电话状态	已使用	android.permission.READ_PHONE_STATE

Case 03

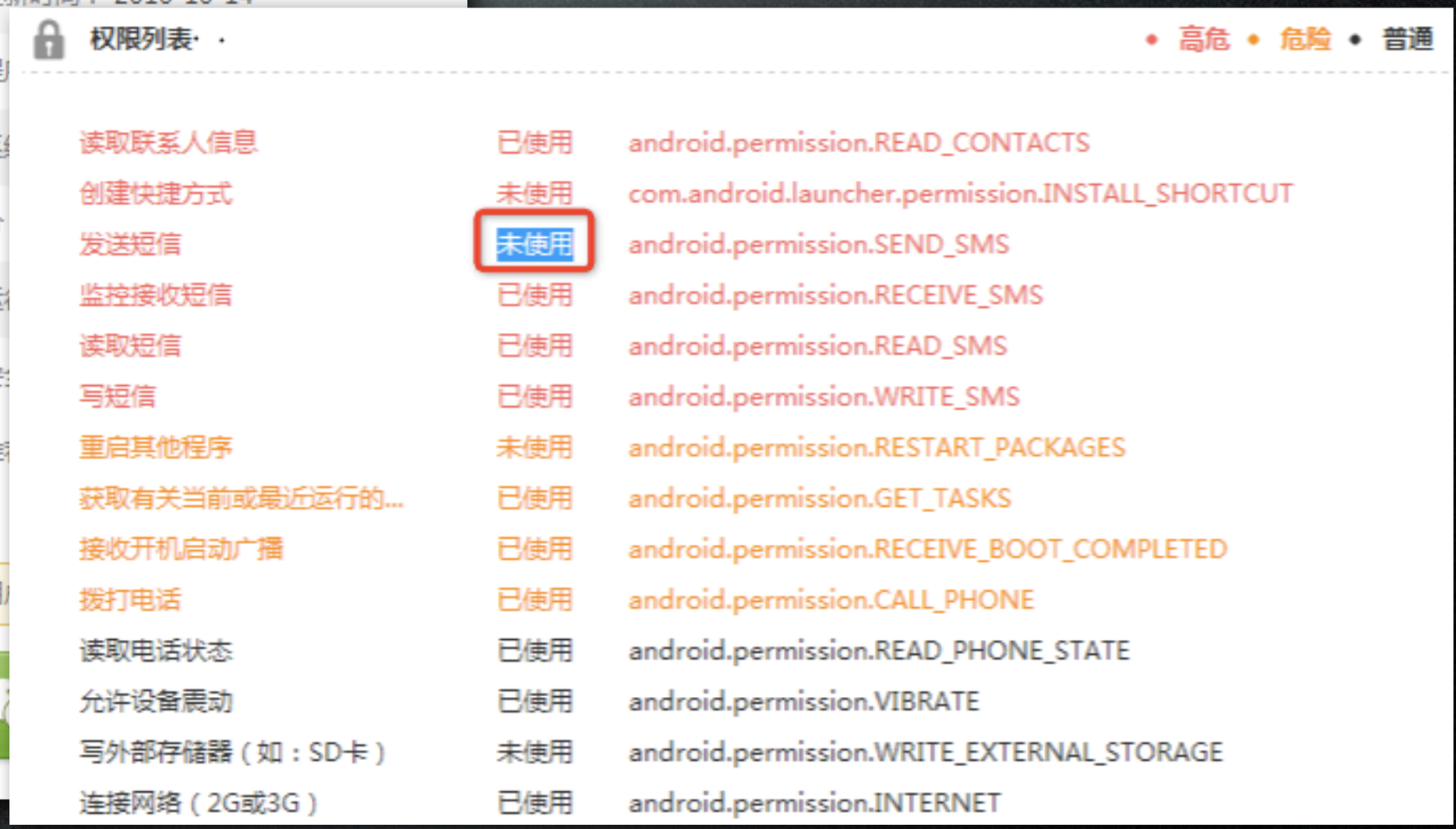
- GSS___duanxinanquan_V1.52_mumayi_e71a7.apk




```
Confirmed reflection:   invoke-virtual  {v1,v2[0],v2[1],v2[2],v2[3],v2[4]},_top/sendMultipartTextMessage        ;
sendMultipartTextMessage(Ljava/lang/String;Ljava/lang/String;Ljava/util/ArrayList;Ljava/util/ArrayList;Ljava/util/ArrayList;)
```

Case 03


GSS - 短信安全 V
GSS - 所有标签
我的会话
私密会话
垃圾信息
系统设置
规则设置
系统工具
帮助信息
分享到：
[一键转帖]
360安全认证
腾讯安全管家认证
软件类型： 免费软件
所属类别： 安全杀毒
更新时间： 2010-10-14


权限列表·
高危
危险
普通
读取联系人信息 已使用 android.permission.READ_CONTACTS
创建快捷方式 未使用 com.android.launcher.permission.INSTALL_SHORTCUT
发送短信 未使用 android.permission.SEND_SMS
监控接收短信 已使用 android.permission.RECEIVE_SMS
读取短信 已使用 android.permission.READ_SMS
写短信 已使用 android.permission.WRITE_SMS
重启其他程序 未使用 android.permission.RESTART_PACKAGES
获取有关当前或最近运行的... 已使用 android.permission.GET_TASKS
接收开机启动广播 已使用 android.permission.RECEIVE_BOOT_COMPLETED
拨打电话 已使用 android.permission.CALL_PHONE
读取电话状态 已使用 android.permission.READ_PHONE_STATE
允许设备震动 已使用 android.permission.VIBRATE
写外部存储器（如：SD卡） 未使用 android.permission.WRITE_EXTERNAL_STORAGE
连接网络（2G或3G） 已使用 android.permission.INTERNET

Performance & Limitation

- Performance
  - Overlapped entrypoint
  - Ad analysis(95% time is consumed for 5% none sense code)
  - Too much disk operation
- Limitation
  - Cast aren’t always present [B. Livshits]

Content

- Background
- aDFAer - Android Data Flow Analyzer
- Janus - Detect Reflection
- Experiment
- Future Work

Future Work

- Methodology
  - TAC IR - > SSA IR
  - Cross activity, service...
  - X-CFA, Object-sensitive...
- Optimize
  - On-demand analysis
  - Shrink memory usage(prune useless node, basic block supported, binary encoding...)
  - Robustness, Efficiency, Extensibility
- More clients

Thanks !

CONTACT us at
info@mobeisecurity.com