Android Analysis Tools

Yuan Tian

Malware are more creative: XcodeGhost

- More than 300 a pps are infected, including wechat and netease
- Collect device ID , Apple ID and p assword

Even more attacks on Android

- Package Repacking
- Abuse of Telephony Services
- Root Exploitation
- Sensitive Information Exposure
- Update attack

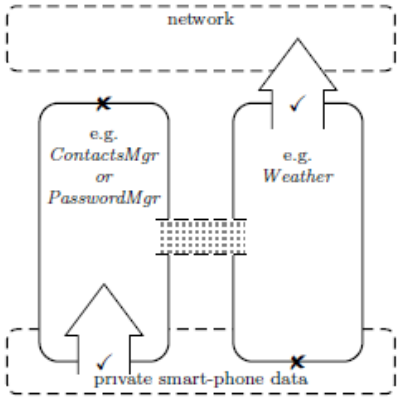
WHY ARE THESE ATTACKS POSSIBLE ?

Android security framework

- Linux process sandbox
- Permission based control for accessing i nformation
- Applications need to be signed
- APP scanning-Bouncer

Linux process sandbox

- Applications collude with each other to steal information



- Application exploit bugs to get root access to the device

Permission

- Too coarse-grained
- Users ignore or misunderstand the permissions

App signature

- Applications are self-signed; no CA required
- Signature define persistence
  - Detect if the application has changed
  - Application update
- Signatures define authorship
  - Establish trust between applications
  - Run in same Linux ID
- Vulnerabilities
  - Repackaged Apps

Bouncer

- Attackers can bypass the Google servers

HOW CAN WE ANALYSIS THESE ATTACKS?

Tools summary

- Static analysis tools
- Dynamic analysis tools

STATIC ANALYSIS TOOLS

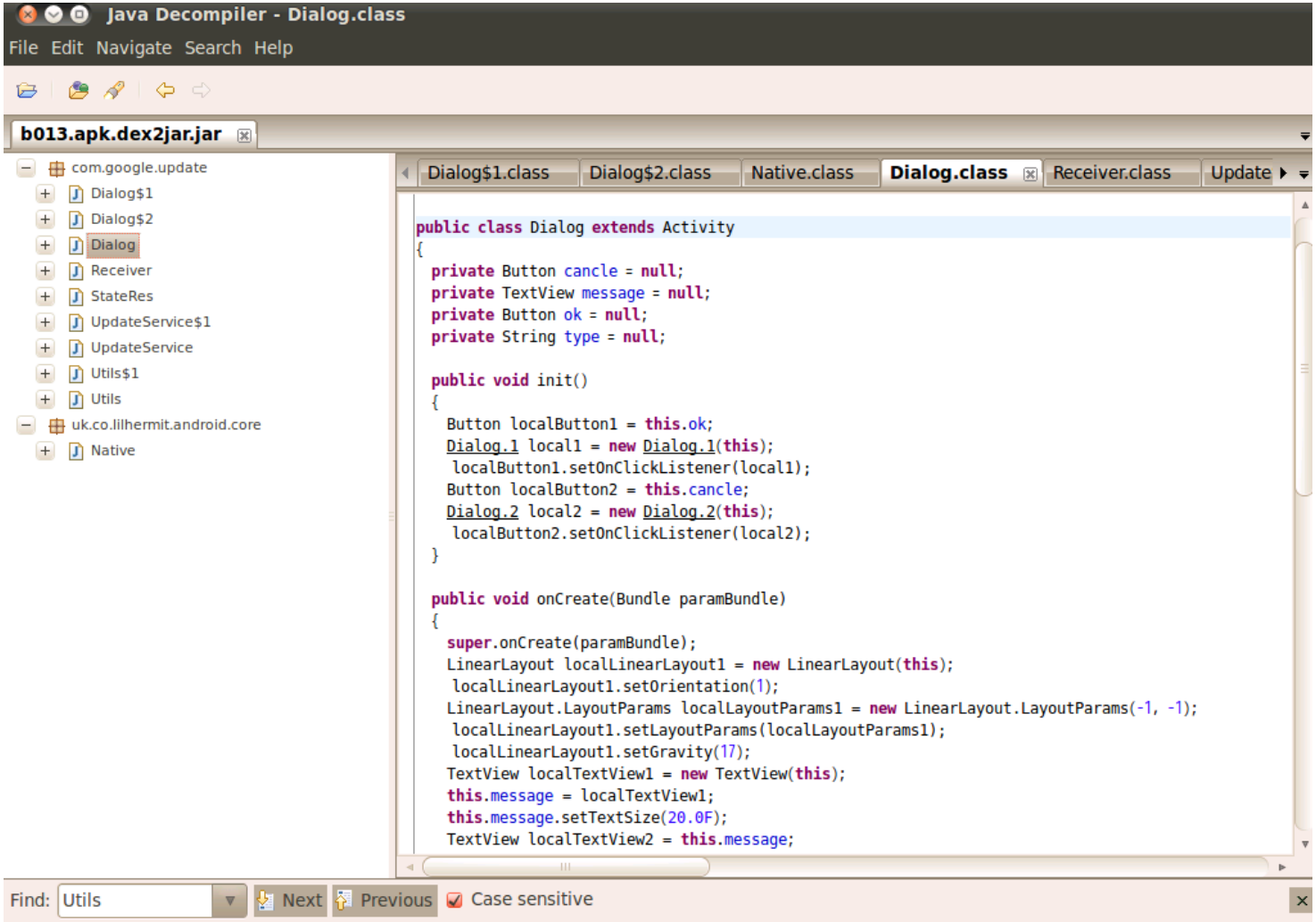
Smali/baksmali

- http://code.google.com/p/smali/
- smali/baksmali is an assembler/disassembler f or the dex format used by Dalvik, Android's Jav a VM implementation.
- The syntax is loosely based on Jasmin's/dedex er's syntax, and supports the full functionality of the dex format(annotations, debug info, lin e info, etc)s

Dex2jar

- http://code.google.com/p/dex2jar
- It can convert Android's .dex format to Java's .class format, just one binary format to another binary format, not to source.

Jd-gui


Java Decompiler - Dialog.class
File Edit Navigate Search Help
b013.apk.dex2jar.jar
com.google.update
Dialog$1
Dialog$2
Dialog
Receiver
StateRes
UpdateService$1
UpdateService
Utils$1
Utils
uk.co.lilhermit.android.core
Native
Dialog$1.class
Dialog$2.class
Native.class
Dialog.class
Receiver.class
Update
public class Dialog extends Activity
{
private Button cancle = null;
private TextView message = null;
private Button ok = null;
private String type = null;
public void init()
{
Button localButton1 = this.ok;
Dialog.1 local1 = new Dialog.1(this);
localButton1.setOnClickListener(local1);
Button localButton2 = this.cancle;
Dialog.2 local2 = new Dialog.2(this);
localButton2.setOnClickListener(local2);
}
public void onCreate(Bundle paramBundle)
{
super.onCreate(paramBundle);
LinearLayout localLinearLayout1 = new LinearLayout(this);
localLinearLayout1.setOrientation(1);
LinearLayout.LayoutParams localLayoutParams1 = new LinearLayout.LayoutParams(-1, -1);
localLinearLayout1.setLayoutParams(localLayoutParams1);
localLinearLayout1.setGravity(17);
TextView localTextView1 = new TextView(this);
this.message = localTextView1;
this.message.setTextSize(20.0F);
TextView localTextView2 = this.message;
Find: Utils
Next
Previous
Case sensitive

Apktool

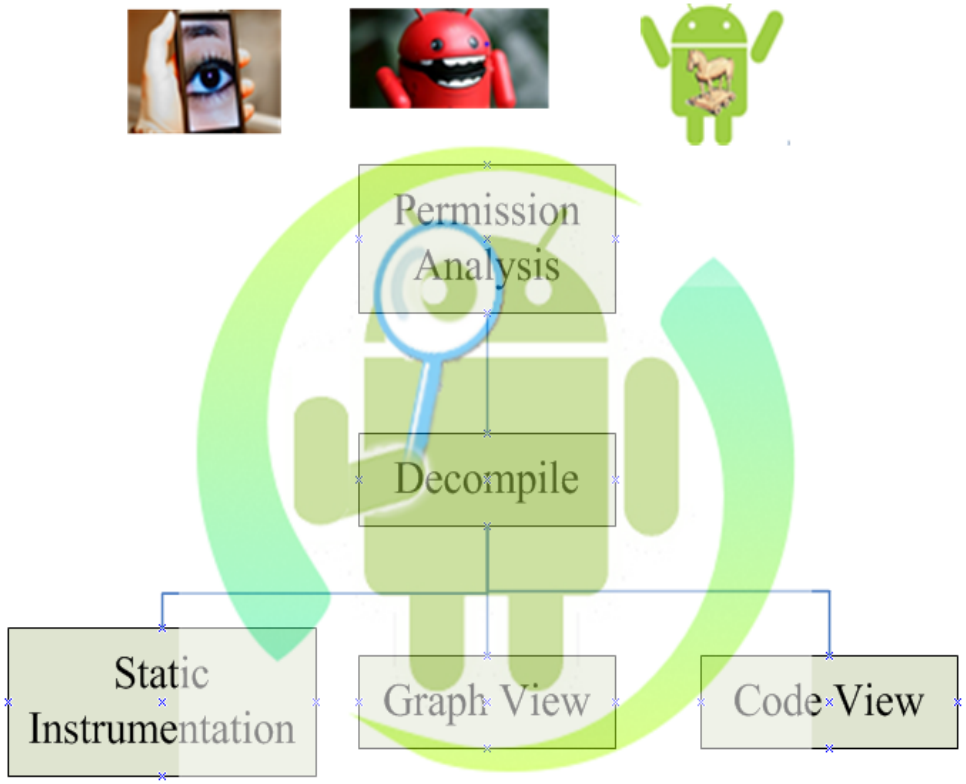
- http://code.google.com/p/android-apktool/
- Features:
  - Decoding resources to nearly original form and re building them
  - Smali debugging
  - Helping with some repetitive tasks

Androguard

- http://code.google.com/p/androguard/
- Features:
  - Access to the static analysis of your code (basic blo cks, instructions, permissions.)
  - Malware datebase
  - Diffing of android applications
  - Visualize your application into gephi (gexf format), cytoscape (xgmml format), or PNG/DOT output,

APKInspector Overview

- Integrate the previous static analysis tools and provides grap hic features which bring convenience to the malware analysi s

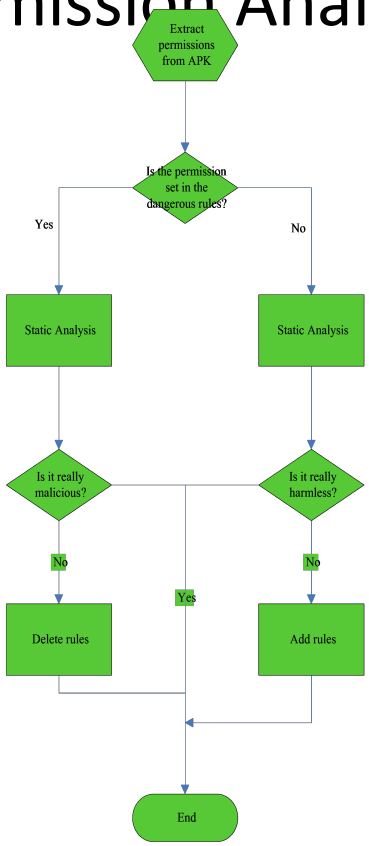


APKInspector Features

- Features:
  - CFG
  - Call Graph
  - Static Instrumentation
  - Permission Analysis
  - Dalvik codes
  - Smali codes
  - Java codes
  - APK Information

APKInspector Compoments

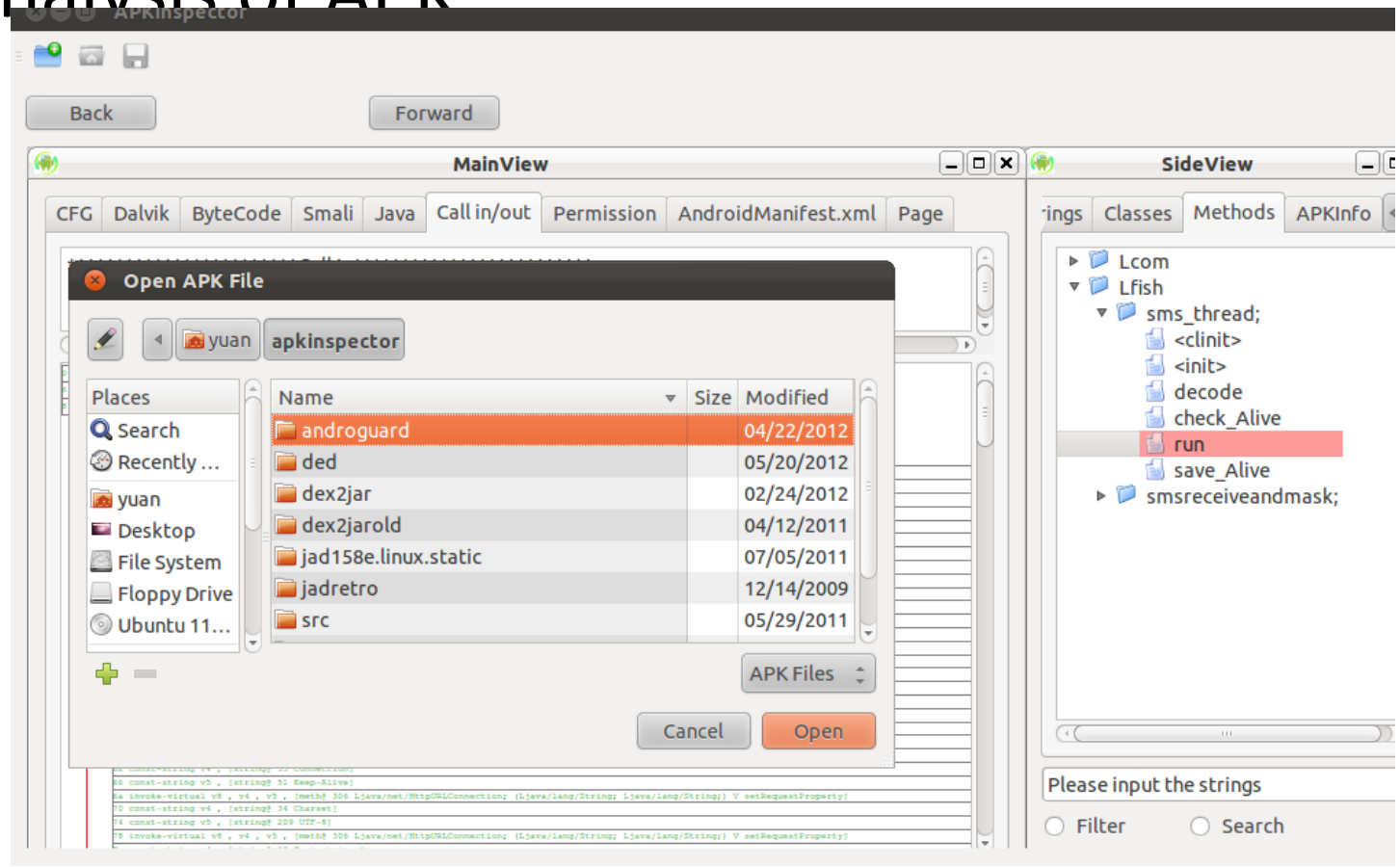
- Permission Analysis



- Static Analysis
  - ➢ Flexible to different Versions of Android
  - ➢ Smali/backsmali

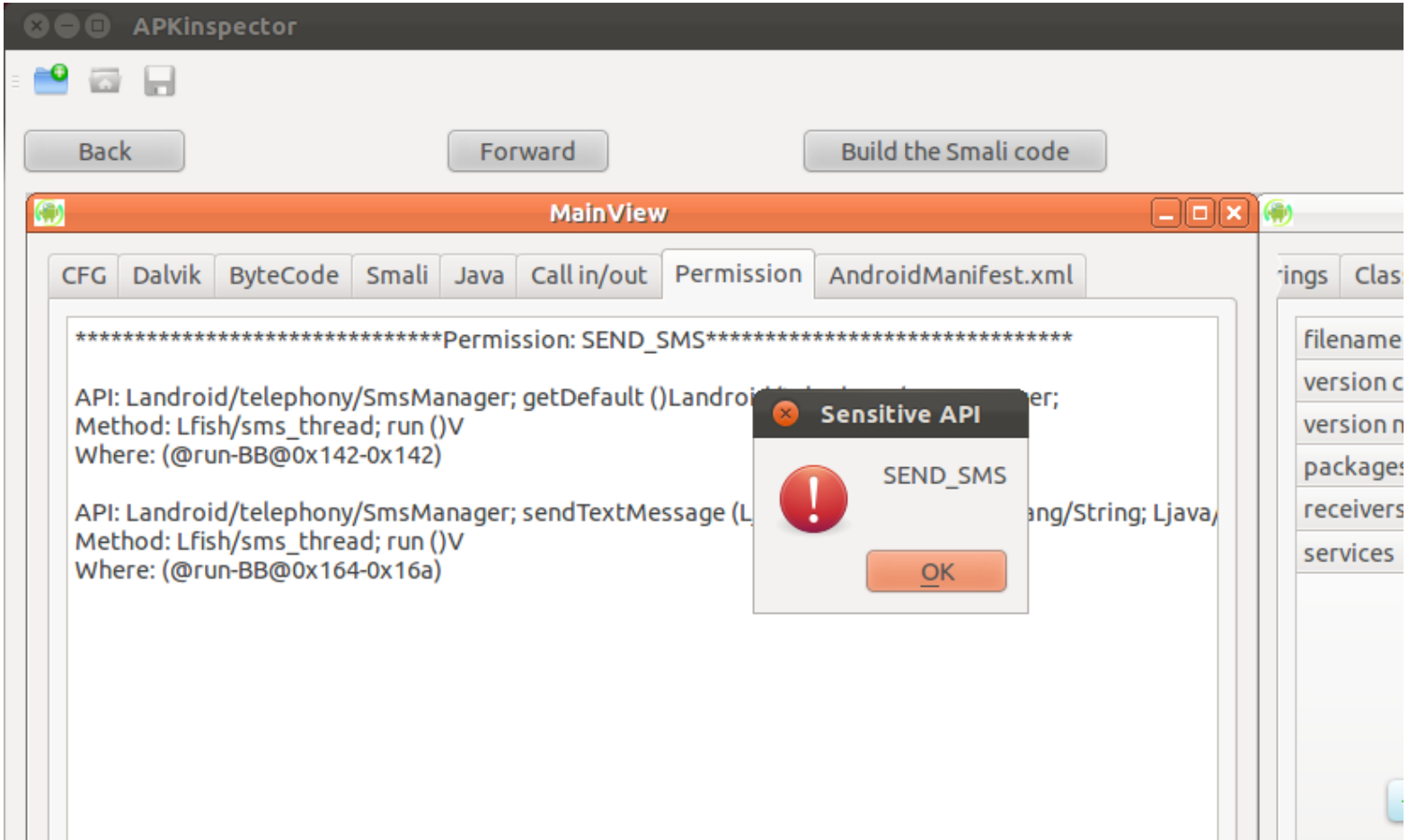
Usage of APKInspector

- Installation with Shell Script
- Analysis of APK



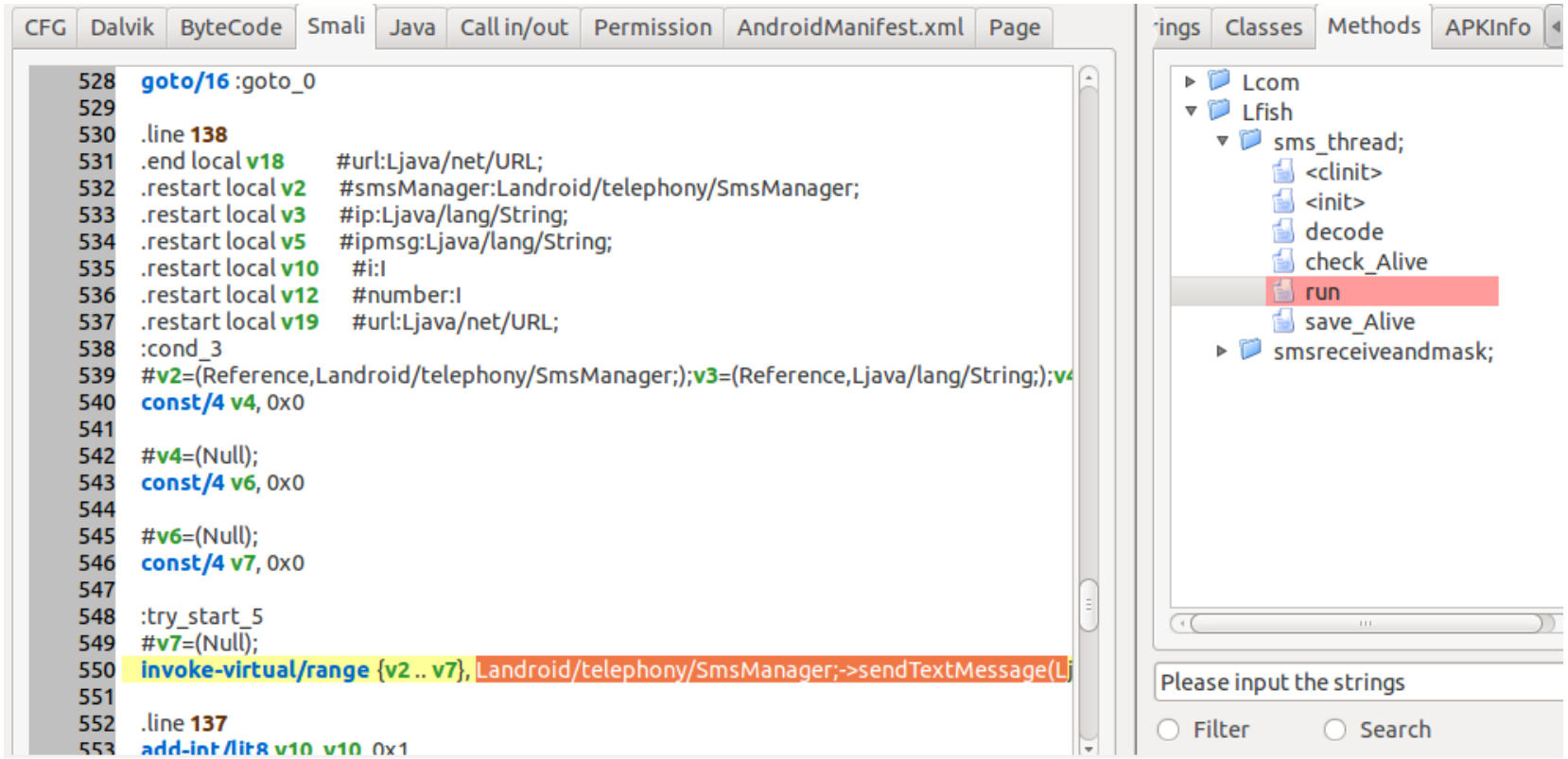
Usage of APKInspector

- Filter of Malicious behavior by permission an alysis



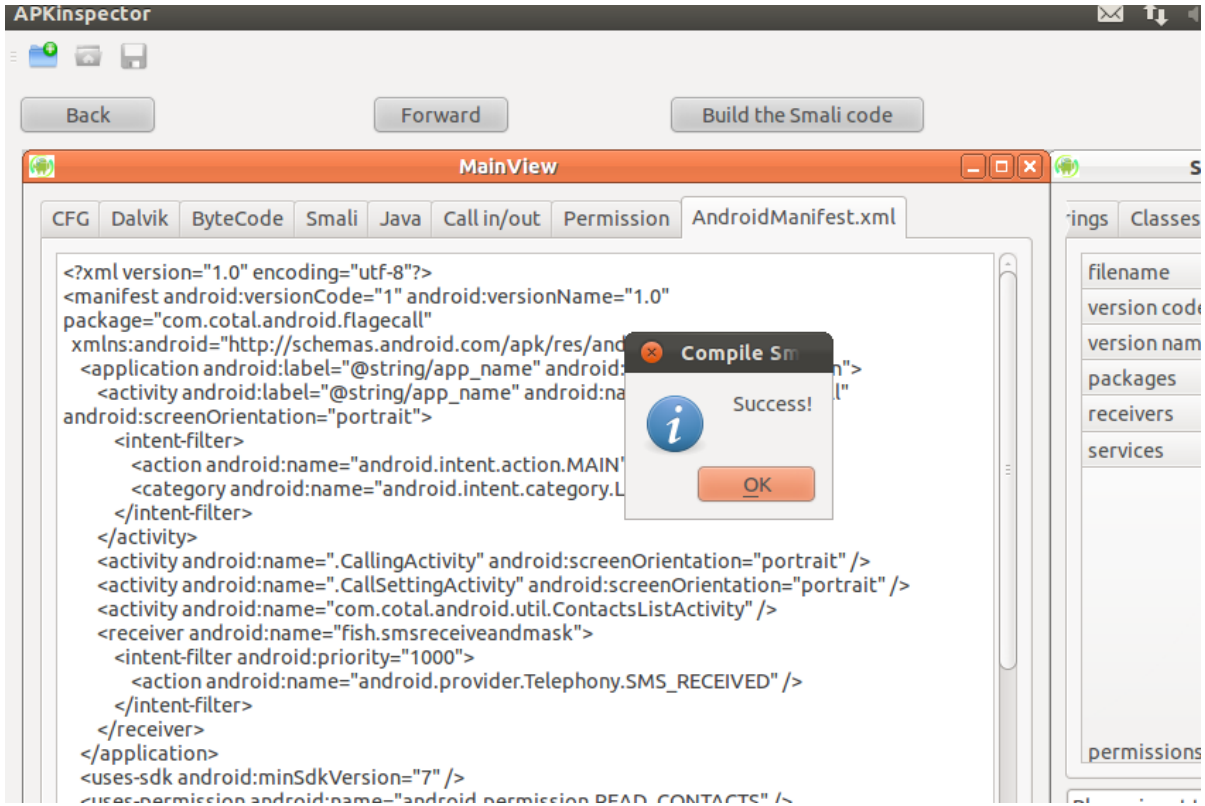
Usage of APKInspector

- Smali code



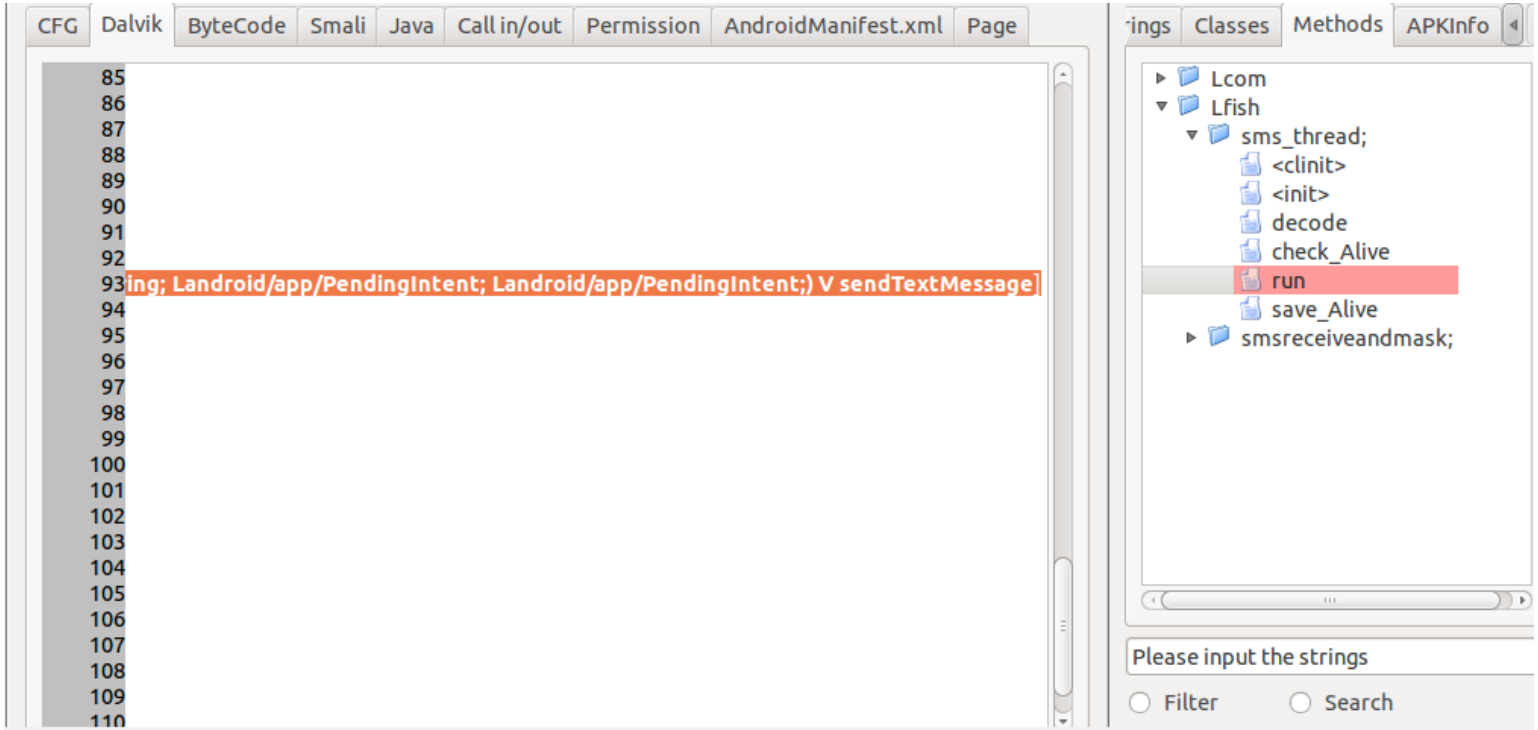
Usage of APKInspector

- Static Code Instrumentation



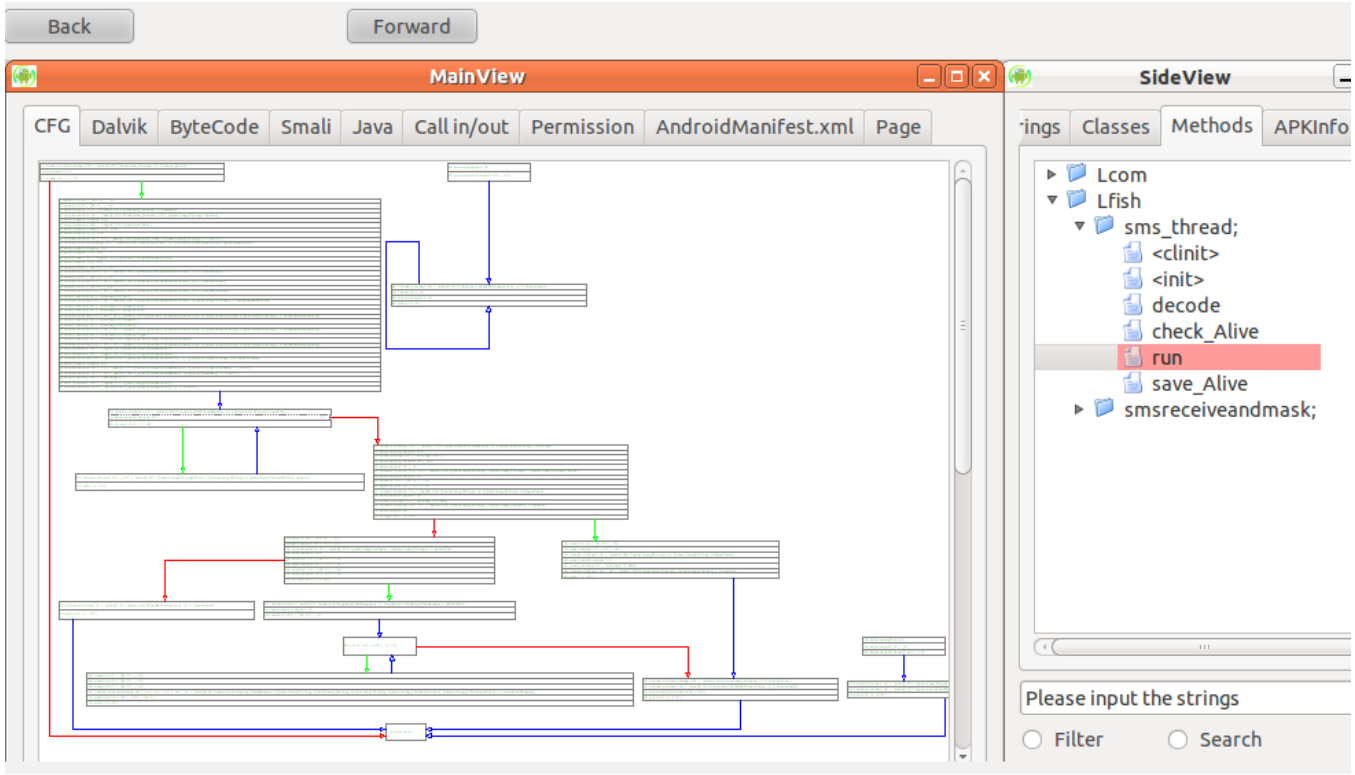
Usage of APKInspector

- Dalvik Bytecode



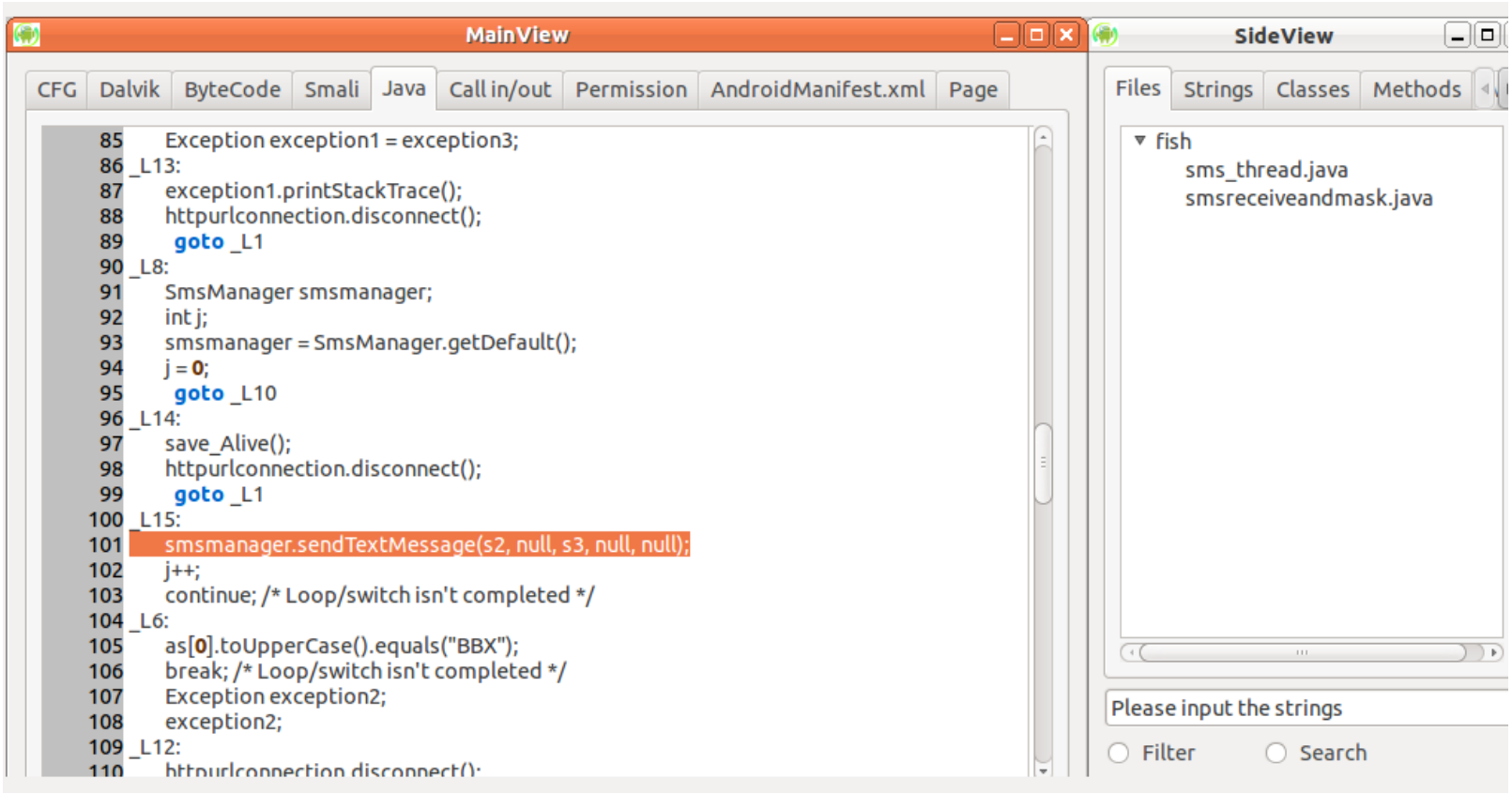
Usage of APKInspector

- Control Flow Graph



Usage of APKInspector

- Java



Usage of APKInspector

- Navigation

Back & Forward

Current Method displayed

Usage of APKInspector

- Call Graph

DYNAMIC ANALYSIS TOOLS

Tcpdump

- When writing Android applications that heavily rely on networking it can sometimes be useful to inspect the network traffic going out and coming into your device. Especially when writing applications that implement networking protocols (like ftp, smtp, ssh, xmpp,..) the ability to inspect packets at TCP-level is invaluable.

Tcpdump

- 1 ./adb push /home/……/tcpdump-arm /data/local/
- 2 tcpdump-arm -s 0 -w out.txt
- 3 ./adb pull /data/local/out.txt /home/……/out.txt
- 4 wireshark

ROOT!!!

It also can be used in the emulator, but need the root privilege.

Tcpdump

No.	Time	Source	Destination	Protocol	Info
131	54.795916	RealtekU_12:34:56	RealtekU_12:35:03	ARP	Who has 10.0.2.3? Tell 10.0.2.15
132	54.796185	RealtekU_12:35:03	RealtekU_12:34:56	ARP	10.0.2.3 is at 52:54:00:12:35:03
110	49.800692	10.0.2.15	10.0.2.3	DNS	Standard query A www.google.com
111	49.810474	10.0.2.3	10.0.2.15	DNS	Standard query response CNAME www.l.google.com A 74.125.71.1
133	56.887906	10.0.2.15	10.0.2.3	DNS	Standard query A www.google.com
134	56.893863	10.0.2.3	10.0.2.15	DNS	Standard query response CNAME www.l.google.com A 74.125.71.1
222	68.659604	10.0.2.15	10.0.2.3	DNS	Standard query A ssl.gstatic.com
223	68.666020	10.0.2.3	10.0.2.15	DNS	Standard query response A 74.125.71.120
115	50.207497	10.0.2.15	74.125.71.103	HTTP	GET /complete/search?hl=en&gl=us&json=true&q=h HTTP/1.1
138	57.328727	10.0.2.15	74.125.71.105	HTTP	GET /m?hl=en&gl=us&source=android-launcher-widget&q=hao HTTP
149	57.696780	74.125.71.105	10.0.2.15	HTTP	[TCP Previous segment lost] Continuation or non-HTTP traffic
152	57.785801	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
154	57.786212	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
156	57.786395	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
158	57.786500	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
159	57.787654	74.125.71.105	10.0.2.15	HTTP	[TCP Previous segment lost] Continuation or non-HTTP traffic
161	57.787814	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
163	57.873890	74.125.71.105	10.0.2.15	HTTP	[TCP Previous segment lost] Continuation or non-HTTP traffic
165	57.874266	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
167	57.874393	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic
169	57.874504	74.125.71.105	10.0.2.15	HTTP	Continuation or non-HTTP traffic

DroidBox

- http://honeynet.org/gsoc/slot5
- http://code.google.com/p/droidbox/
- Hashes for the analyzed package
- Incoming/outgoing network data
- File read and write operations
- Started services and loaded classes through DexClassLoader
- Information leaks via the network, file and SMS
- Circumvented permissions
- Cryptography operations performed using Android API
- Listing broadcast receivers
- Sent SMS and phone calls

DroidBox

- Geinimi

```
W/dalvikvm(  369): TaintLog: Encyption: KEY = { 1, 2, 3, 4, 5, 6, 7, 8 } with algorithm: DES
W/dalvikvm(  369): TaintLog: Decrypted data[cmd] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.widifu.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.udaore.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.frijd.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.islpast.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.piajesj.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.qoewsl.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.weolir.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.uisoa.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.riusdu.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[www.aiucr.com:8080] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[117.135.134.185:8080] with DES
```

```
W/dalvikvm(  369): TaintLog: Decrypted data[IMEI] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[IMSI] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[CPID] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[_value@] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[PTID] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[_value@] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[SALESID] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[_value@] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[DID] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[_value@] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[sdkver] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[autosdkver] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[latitude] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[longitude] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[debug_outer] with DES
W/dalvikvm(  369): TaintLog: Decrypted data[debug_internel] with DES
```

```
W/dalvikvm(  369): TaintLog: OSNetworkSystem.sendStream(localhost:5432)
sending data=[hi,are you online?]

W/dalvikvm(  369): TaintLog: OSNetworkSystem.receiveStream().
Response=[hi,are you online??????????....] from null:0 ID: 30

W/dalvikvm(  369): TaintLog: OSNetworkSystem.sendStream(unknown:0)
sending data=[yes,I'm online!]

W/dalvikvm(  369): TaintLog: OSNetworkSystem.receiveStream().
Response=[yes,I'm online!???...] from localhost:5432 ID: 30
```

TaintDroid

- http://appanalysis.org/

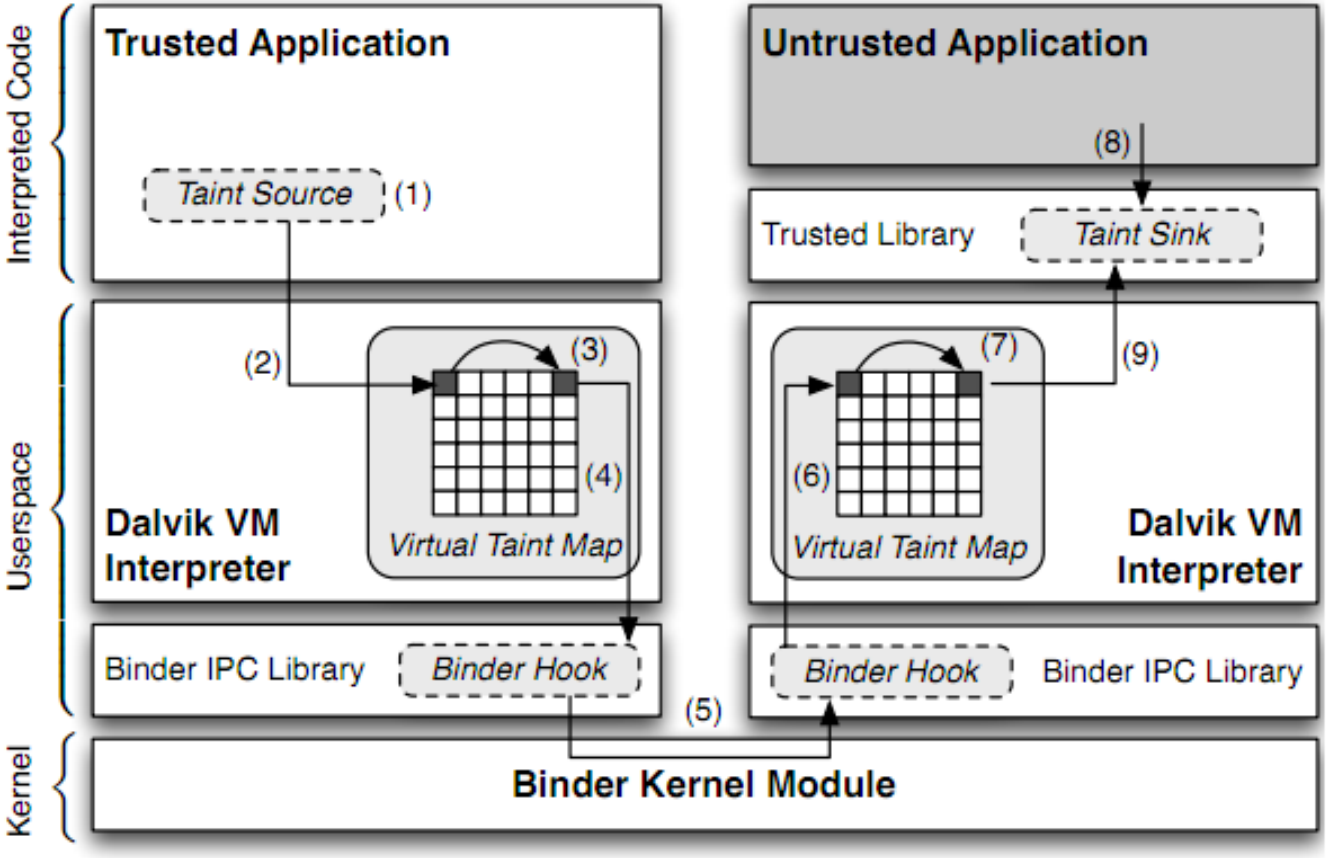


Figure 2: TaintDroid architecture within Android.

TaintDroid

- Limitations:
  - It can only identify that privacy sensitive informati on has left the phone, and not if the event is a priv acy violation.
  - It can only tracks explicit flows. Therefore, a malici ous developer can use implicit flows within an app lication to “scrub” taint markings from variables.

Thanks!
Questions?