

# Automated Forensic Analysis of Mobile Applications on Android Devices

Xiaodong Lin, Ting Chen, Tong Zhu, Kun Yang, Fengguo Wei

Xiaodong Lin, PhD, IEEE Fellow

Associate Professor

Department of Physics and Computer Science

Wilfrid Laurier University, Canada

[xlin@wlu.ca](mailto:xlin@wlu.ca)

# Android Digital Forensics is Important

THE BLOG 02/12/2016 05:31 pm ET | Updated Feb 12, 2017

## The Cell Phone Evidence in Adnan Syed's Case Illustrates a Depressingly Common Problem

By Kevin Sali



BALTIMORE SUN VIA GETTY IMAGES



The hearing is over for Adnan Syed, whose 2000 murder conviction came to national attention through the *Serial* podcast. The judge is now deciding whether the evidence presented at that hearing warrants Syed's request for a new trial. The request centers on two issues — the alleged ineffectiveness of Syed's original trial lawyer Cristina Gutierrez in failing to contact a potential alibi witness, and alleged reliability issues with the cell phone evidence used to place Syed near the location where the victim's body was found.

Some background as to that second question — using historical call records to trace a subscriber's location is a relatively recent development. The basic theory is that cell phones making or receiving calls will typically connect up with nearby cell towers, so tracking which towers were communicating with a particular cell phone during a given period can provide information about where that phone was during that period. It's been a controversial topic, with prosecutors and defense lawyers frequently sparring over the reliability of this type of evidence.

# Common Techniques

Data storage questions: **what** is the information stored (e.g., GPS); **where** is the information stored (e.g., file path); and **how** the information is stored (e.g., the structure of a database table).

*Finding data*



[1] N. Scrivens, X. Lin. "ndroid digital forensics: data, extraction and analysis". ACM TUR-C 2017, Shanghai, China.

[2] C. Anglano. "Forensic analysis of whatsapp messenger on android smartphones", Digital Investigation. 11 (2014): 201-213.

# Common Techniques - Limitations



- Hard to trigger all interesting program paths. Consequently, some behaviors of a mobile app may not be discovered by dynamic analysis.
- Nontrivial to identify what information is stored and how it is stored. For example, a file generated by a mobile application whose content is encoded or whose format is unknown needs considerable efforts to analyze.
- Hard to automate dynamic analysis given a large number of applications due to the differences in runtime environments as well as increasingly difficult to keep up to speed with new applications

[1] N. Scrivens, X. Lin. "Android digital forensics: data, extraction and analysis". ACM TUR-C 2017, Shanghai, China.

[2] C. Anglano. "Forensic analysis of whatsapp messenger on android smartphones", Digital Investigation. 11 (2014): 201-213.

# Outline

- **Motivating Example**
- Fordroid
- Evaluation
- Conclusions and Future directions

# Motivating Example

Agilebuddy

Game app

703KB

13 packages

7 components




80 classes

559 functions

**How information is written to files?**

**Manual reverse engineering is burdensome!**

```
26  h.f = 8192;
124 if(!arg6){
125     try{
126         if(h.g.length() < h.f){
127             }
128         else if(h.e != null){
129             goto label_11;
130         }
131         goto label_9;
133 label_11:
134     if(!Environment.getExternalStorageState().equals(`mounted`)){
135         goto label_9;
136     }
137     ...
138     v0_1 = h.a(new Date(), `yyyy-MM-dd` + `.log`);
139     File v1 = new File(Environment.getExternalStorageDirectory(),
140         h.e.getPackageName() + `/logs/`);
141     v4 = new File(v1, v0_1);
142     ...
162     v2 = new FileWriter(v4, true);
163     ...
171     v2.write(v0_1);
```



(a) Code snippet in `function c(), class h, package com.uucunadsdk.h`

# Motivating Example (cont'd)

To reproduce such behavior

- c() should be invoked.
- arg6 (Line 124) should be false.
- **h.g.length() should be no small than 8192 (Line 126).**
- h.e should not be null (Line 128).
- an sdcard should be mounted (Line 134).

```
26  h.f = 8192;
124 if(!arg6){      arg6 should be false
125     try{
126         if(h.g.length() < h.f){ h.g.length() should be no small than 8192
127     }
128     else if(h.e != null){      h.e should not be null
129         goto label_11;
130     }
131     goto label_9;
133 label_11:
134     if(!Environment.getExternalStorageState().equals(`mounted`)){ an sdcard should be mounted
135         goto label_9;
...
138     v0_1 = h.a(new Date(), ``yyyy-MM-dd`` + ``.log``);
139     File v1 = new File(Environment.getExternalStorageDirectory(),
        h.e.getPackageName() + ``/logs/``);
140     v4 = new File(v1, v0_1);
...
162     v2 = new FileWriter(v4, true);
...
171     v2.write(v0_1);
```

(a) Code snippet in function c(), class h, package com.uucunadsdk.h

# Motivating Example (cont'd)

To satisfy Line 126 → 126

if(h.g.length() < h.f){

h.f=8192

h.g.length() should be no small than 8192

- h.g stores exception info.
- h.a() produces exception info
- 1 run of h.a() appends no longer than 100 bytes to h.g.
- We need to trigger **at least 80 exceptions** before file creation

**Dynamic analysis is difficult to trigger the program path to the code of interest!**

```
72 v0.append(arg6).append("l").append(h.a(new Date(),  
    "yyyy-MM-dd HH:mm:ss")).append("l").append(v1[v5].  
    getClassName()).append("l").append(v1[v5].getMethodName()).  
    append("l").append(v1[v5].getLineNumber());  
73 h.g.append(v0.toString()).append("l").append(arg8).append("\n");
```

(b) Code snippet in function a(), class h, package com.uucunadsdk.h

```
166 catch(UnknownHostException  
...  
169 try{  
170 label_86:  
171 h.a(this.a, v0.toString());
```

(c) Code snippet in function b(), class m, package com.uucunadsdk.c

**Failed to create  
the file using  
dynamic analysis**



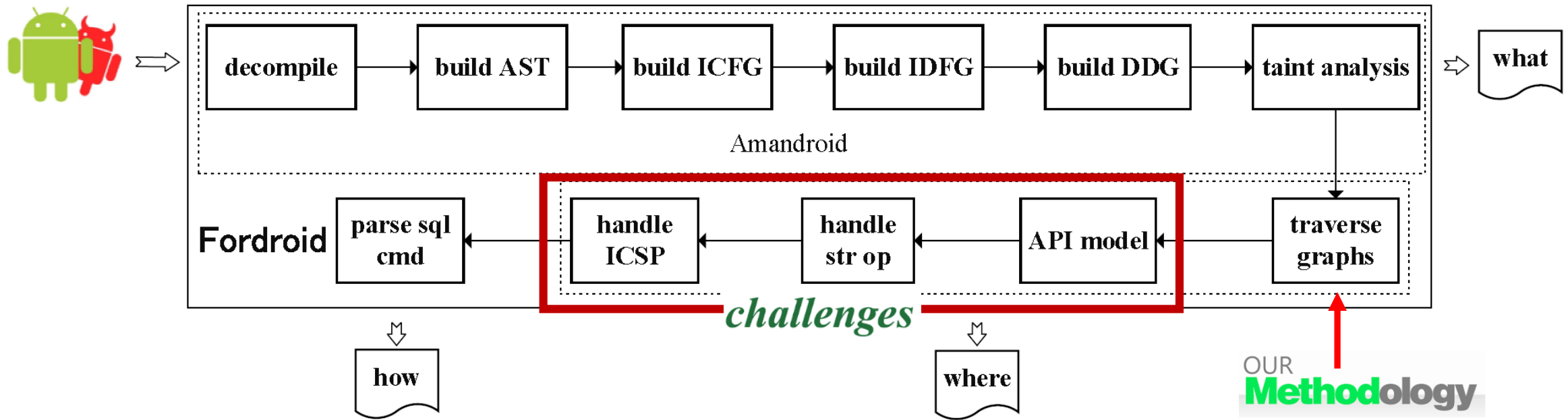
# Outline

- Motivating Example
- **Fordroid**
- Evaluation
- Conclusions and Future directions

# Fordroid

## Goals

- Android app forensic analyzer
- **Full automatic**
- Identify what and where information written in local storage



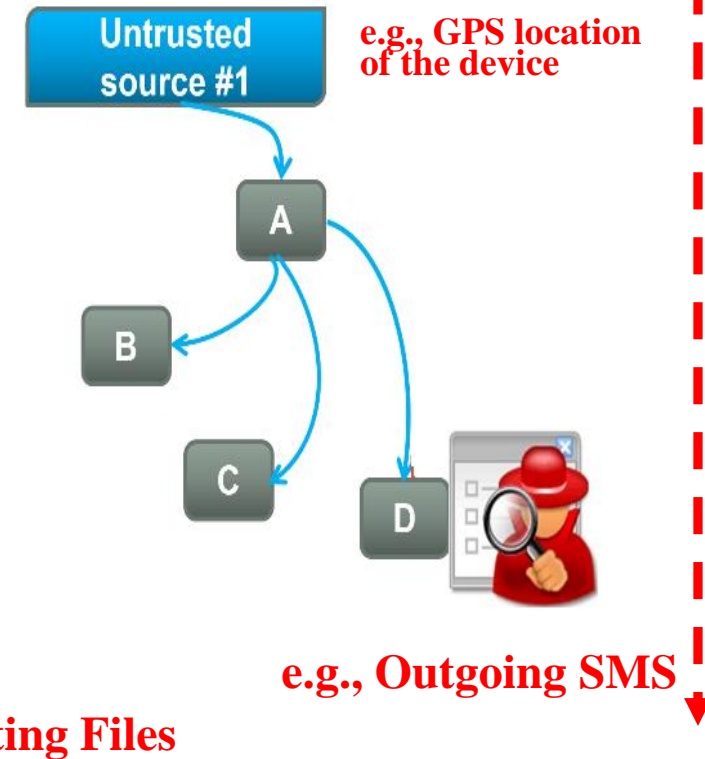
# Fordroid

## OUR Methodology

### What info is stored **locally**?

- Taint analysis
  - **Taint source** (e.g., `getLastKnownLocation()`): Originally, associate taint marker with untrusted input as it enters the program. Here, we mark **any data stored locally**.
  - **Taint sink** (e.g., `Write()`): Originally, mark sensitive sinks and report vulnerabilities when tainted strings are passed to these sinks. Here, we report **local data storage activities**, such as written into a file.
  - **Taint propagation**: Propagate markers when string values are copied or concatenated
- Enrich Amandroid<sup>[3]</sup> (i.e., source, sink)

### Taint analysis



[1] D. Denning and P. Denning. "Certification of programs for secure information flow". Communication of the ACM, 1977.

[2] J. Newsome and D. Song. "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software." Network and Distributed System Security Symposium (NDSS), 2005

[3] F. Wei, S. Roy, X. Ou and Robby, "Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps", ACM CCS, Scottsdale, AZ, USA, 2014.

# Fordroid

## Where is info stored?

Android provides the following four mechanisms for storing and retrieving data:

1. Preferences: an Android lightweight mechanism to store and retrieve key-value pairs of primitive data types. Typically used to keep state information and shared data among several activities of an application.
2. Files;
3. Databases;
4. Network: for example, cloud-based storage.

Local data  
storage

Cloud data  
storage

# Fordroid

Where is info stored **locally**?

- Modes
  - Mode 1: SharedPreferences.
  - Mode 2: Databases.
  - Mode 3: Files.
- Challenges
  - Challenge 1: Inter-component string propagation.
  - Challenge 2: String operations.
  - Challenge 3: API invocations.

# Fordroid

## Mode 1

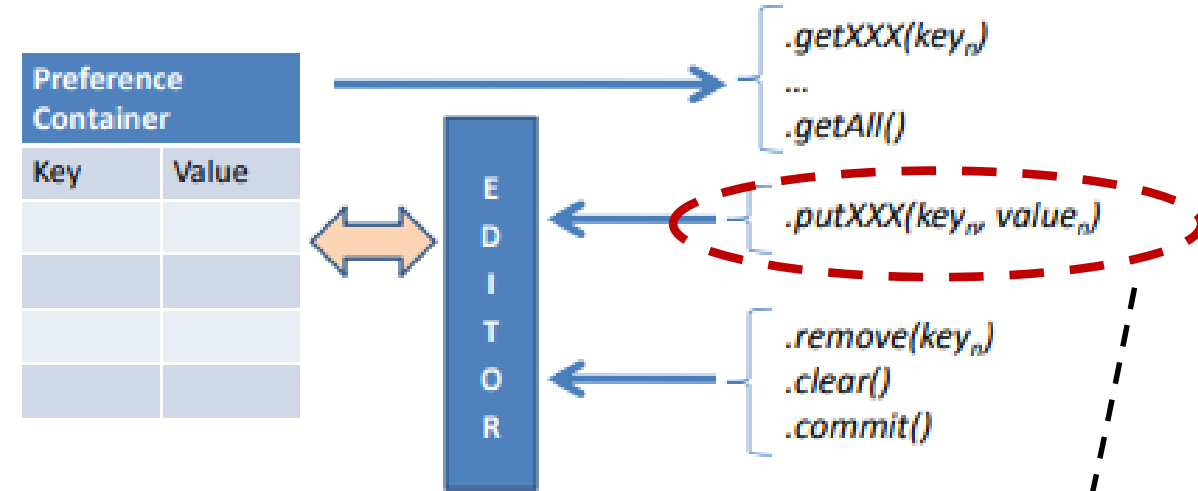
Using Preferences API calls

### SharedPreferences

Backtrack from taint sink to where defines  
SharedPreferences.

- Taint sink: Line 443.
- Caller: v0\_1 defined at Line 441.
- Looking for definition site of v2.
- Defsite: Line 423.
- 1st parameter: "message\_prefs".

```
423 SharedPreferences v2 = arg10.getSharedPreferences  
    ("message_prefs", 0);  
...  
441 SharedPreferences$Editor v0_1 = v2.edit();  
442 v0_1.remove(v3);  
443 v0_1.putString(v3, arg11.a());
```



# Fordroid

## Mode 2

### Databases

Find table name in sink, find database name by backtracking.

- Taint sink, Line 45.
- Table name, “downloads”, Line 45.
- Caller of insert, v0, Line 45.
- Defsite of v0, Line 38.
- Search AST to find this.a, Line 38, whose constructor a(), Line 8.
- Invocation of superclass’s constructor, Line 9.
- Database name is 2nd parameter, “downloads”.

```
38 SQLiteDatabase v0 = this.a.getWritableDatabase();
39 ContentValues v1 = new ContentValues();
40 v1.put("url", arg5.a);
41 v1.put("file", arg5.b);
42 v1.put("size", Integer.valueOf(arg5.c));
43 v1.put("total_size", Integer.valueOf(arg5.d));
44 v1.put("state", Integer.valueOf(arg5.e));
45 v0.insert("downloads", null, v1);
```

(a) Code snippet in function a(), class b, package com.kuguo.a

```
7 class a extends SQLiteOpenHelper {
8     a(Context arg4) {
9         super(arg4, "downloads", null, 1);
10    }
```

(b) Code snippet in class a, package com.kuguo.a

# Fordroid

## Mode 2

### Databases (cont'd)

#### Identify the Structure of Database Table.

- monitor the API, `execSQL()` for executing SQL commands and then extract SQL commands from its parameters.
- find the SQL command for creating database table by looking for the keyword "**CREATE TABLE**".
- parse the SQL command to retrieve the table name and the name and type of each column.

```
12 public void onCreate(SQLiteDatabase arg2) {  
13     arg2.execSQL("CREATE TABLE downloads(  
        _id INTEGER PRIMARY KEY, url TEXT, file TEXT,  
        size INTEGER, total_size INTEGER, state INTEGER);");  
}
```

Figure 5: A code snippet from class `a`, package `com.kuguo.a`

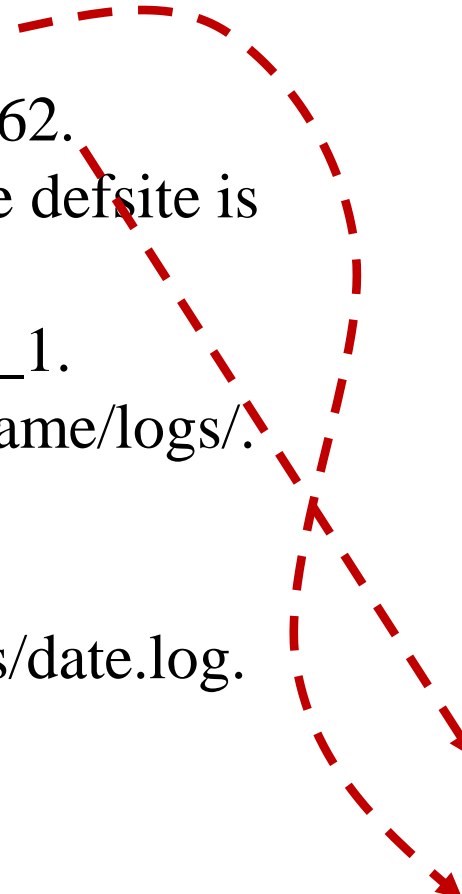


# Fordroid

## Mode 3

### Files

Backtracking from taint sinks.

- Taint sink, Line 171.
  - Defsite of v2, Line 162.
- 1st parameter v4, whose defsite is Line 140.
- Backtrack v1 and v0\_1.
  - v1 part: sdcard/pkgname/logs/.
  - v0\_1 part: date.log.
  - Complete file name:  
sdcard/pkgname/logs/date.log.
- 

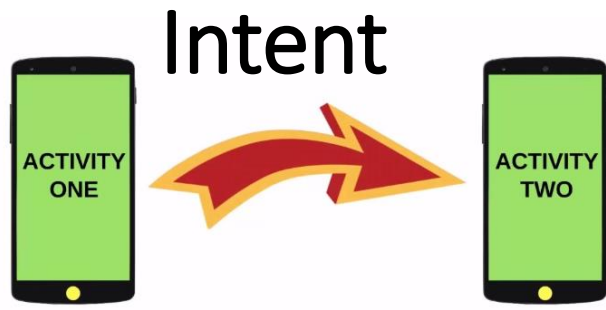
```
26  h.f = 8192;
124 if(!arg6){
125     try{
126         if(h.g.length() < h.f){
127             }
128         else if(h.e != null){
129             goto label_11;
130         }
131         goto label_9;
133 label_11:
134         if(!Environment.getExternalStorageState().equals("`mounted`")){
135             goto label_9;
136         }
137         ...
138         v0_1 = h.a(new Date(), "`yyyy-MM-dd`" + `.log`);
139         File v1 = new File(Environment.getExternalStorageDirectory(),
140             h.e.getPackageName() + `/logs/`;
141         v4 = new File(v1, v0_1);
142         ...
143         v2 = new FileWriter(v4, true);
144         ...
145         v2.write(v0_1);
```

(a) Code snippet in function c(), class h, package com.uucunadsdk.h

# Fordroid



## Challenge 1: Inter-Component String Propagation



- Reuse Amandroid to handle ICC.
- Model two functions:
  - `putStringExtra()` packs a string in an Intent - `Intent.putExtra()`
  - `getStringExtra()` extracts a string from an Intent - `Intent.getStringExtra()`

## Challenge 2: string operations

- append, substring, index, etc.
- Precise string analysis: powerful, high overhead <sup>[1]</sup>.
- Append is most-widely used.
- Lightweight string analysis by modelling the API `StringBuilder.append()`.

```
138     v0_1 = h.a(new Date(), ``yyyy-MM-dd` + `.log``;  
139     File v1 = new File(Environment.getExternalStorageDirectory(),  
        h.e.getPackageName() + `/logs/`;  
140     v4 = new File(v1, v0_1);  
...  
162     v2 = new FileWriter(v4, true);  
...  
171     v2.write(v0_1);
```

[1] D. Li, Y. Lyu, M. Wan, W. G. J. Halfond. String Analysis for Java and Android Applications. The 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). September 2015.



## Challenge 3: API Invocations

- Some APIs are frequently invoked, whose return values are parts of file names
- Date() (Line 138),  
getExternalStorageDirectory() (Line 139) and  
getPackageName() (Line 139)
- models common APIs

```
138     v0_1 = h.a(new Date(), ``yyyy-MM-dd`` + ``.log``);
139     File v1 = new File(Environment.getExternalStorageDirectory(),
                h.e.getPackageName() + ``/logs/``);
140     v4 = new File(v1, v0_1);
...
162     v2 = new FileWriter(v4, true);
...
171     v2.write(v0_1);
```

# Outline

- Motivating Example
- Fordroid
- **Evaluation**
- Conclusions and Future directions

# Evaluation

Table 1: Analysis results of 100 Android applications

category	#APKs	# comp.	time (min)	# paths	# paths to storage			where		# APKs with paths	# APKs write storage
					sp	db	file	suc.	fa.		
comm.	26	978	1,827	310	27	0	7	33	1	11	6
enter. & game	26	278	207	422	196	18	8	221	1	11	8
news & info.	24	715	902	360	30	4	10	38	6	16	10
tool	24	870	924	1,221	163	0	6	166	3	18	12
<b>total</b>	100	2,841	3,860	2,313	416	22	31	458	11	56	36
<b>ave.</b>	/	28.4	38.6	23.1	41.6	2.2	3.1	45.8	1.1	/	/

**Efficient: 38min/app.**

**Effective in locating where information is stored locally:  $458/469 = 98\%$ .**

**Successfully reveals the structure of all (i.e., 22) database tables.**

# Evaluation (cont'd)

Failed cases in identifying data storage location

11 paths (2%).

Reason1: string operations, 3 paths.

Reason 2: input dependency, 8 paths.

**R1:** String operations

hashCode(), substring() are not handled.

```
12 public static final String a(Context arg10, String arg11) {  
...  
21     String v1 = ...  
...  
33     String v2 = String.valueOf(arg11.hashCode()) + "." +  
...  
...  
...  
35     v6 = v1 + v2;  
...  
42     return v6;}
```

(c) Generate file path in in class b, package com.yulong.d

# Evaluation (cont'd)

Failed cases in identifying data storage location

**R2:** Input dependency

doInBackground() is a callback function for executing asynchronous tasks which receive inputs.

```
34  protected Bitmap doInBackground(String[] arg15) {  
35      Bitmap v11;  
36      this.imageUrl = arg15[0];  
...  
44      String v3 = StringUtil.encodeByMD5(this.imageUrl);  
45      File v5 = new File(...);  
46      this.imagefile = new File(v5, v3);  
...  
51      BitmapUtil.copy(v9, this.imagefile);}
```

(b) Create file in class LruImageAsyncTaskForAcitivity,  
package com.mofang.radish.utils

# Outline

- Motivating Example
- Fordroid
- Evaluation
- **Conclusions and Future directions**



# Conclusions and Future directions

Fully automated forensic analysis tool for Android apps using static analysis.

Unveil what, where and how information is stored in local storage.

Overcome three technical challenges.

Fordroid is effective and efficient.

Future directions:

Stronger string analysis ability.

Reverse engineering file formats.



*Thanks for your attention!*