Do Android Taint Analysis Tools Keep Their Promises?

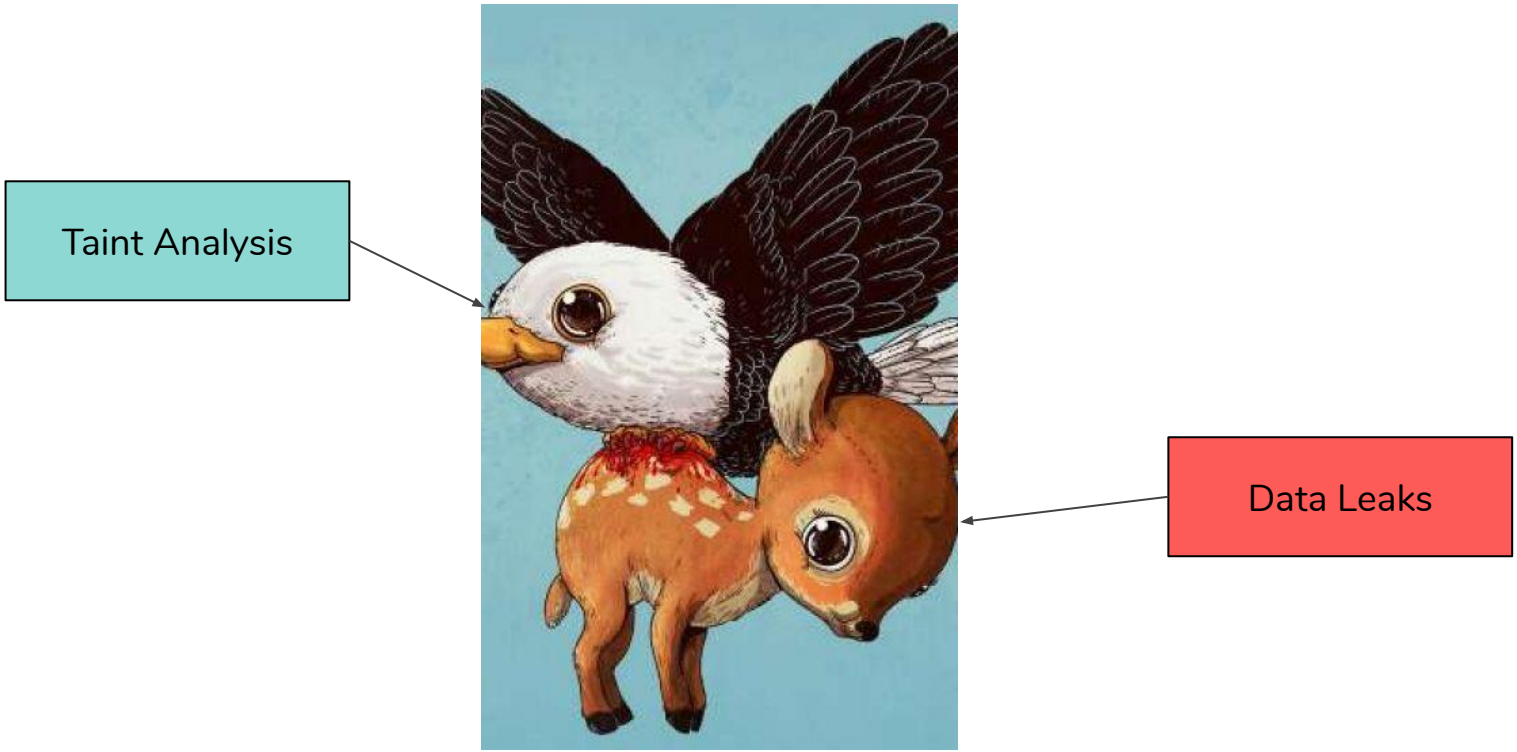
Pauck, F., Bodden, E., Wehrheim, H.

Overview

- Background
  - Taint Analysis
  - Benchmarks
- Approach
  - ReproDroid
    - AQL (Android App Analysis Query Language)
    - AQL-System
    - Brew (Benchmark Refinement and Execution Wizard)
- Research Questions
- Setup
  - Tool Selection
  - Benchmark Selection
  - Experiments
- Results
  - Answering Research Questions
  - Threats to Validity

Background

Taint Analysis



Taint Analysis

- Aliasing
  - Same memory location shared by many variables (reference types).
  - Taint should be carried to all other variables
- Static Fields
  - Declared on a type, not an instance.
  - Treat static fields differently
- Lifecycle and Callbacks
- Inter-Component Communication
  - A leak can originate in a class and end in another.
  - Android allows for inter-component or inter-app communication (ICC/IAC)
  - This allows tainted data to be propagated between components and apps
- Analysis Abstraction and Algorithms
  - The taint analysis may or may not support different łsensitivitiesž, such as flow, context, path, field, object and/or thread-sensitivity
- Reflection
  - Allows to invoke methods (or access fields) through dynamically generated strings.
  - Resolve these strings to reliably detect taint flows.

Taint Analysis

```
public class MainActivity extends Activity {

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

        TelephonyManager mgr = (TelephonyManager) this.getSystemService(TELEPHONY_SERVICE);
                SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage("+49 1234", null, mgr.getDeviceId(), null, null); //source, sink, leak
    }
}
```

Taint Analysis

```
public class PrivateDateLeakage extends Activity {
        private User user = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_private_date_leakage);
    }

    @Override
        protected void onRestart(){
                super.onRestart();
                EditText usernameText = (EditText)findViewById(R.id.username);
                EditText passwordText = (EditText)findViewById(R.id.password);

                String uname = usernameText.toString();
                String pwd = passwordText.getText().toString(); //source

                user = new User(uname, pwd);
        }
```

```
public void sendMessage(View view){
        if(user != null){
                String password = getPassword();
                String obfuscatedUsername = "";
                for(char c : password.toCharArray())
                        obfuscatedUsername += c + "_";

                String message = "User: " + user.getUsername() + " | Pwd: " + obfuscatedUsername;
                SmsManager smsmanager = SmsManager.getDefault();
                Log.i("TEST", "sendSMS"); //sink
                smsmanager.sendTextMessage("+49 1234", null, message, null, null); //sink, leak
        }
}

private String getPassword(){
        if(user != null)
                return user.getPwd().getPassword();
        else{
                Log.e("ERROR", "no password set");
                return null;
        }
}
```

Benchmarks

- Used to evaluate Android app analysis tools
- A collection of apps containing certain features
- DroidBench, ICC-Bench
  - Imprecise ground truth
- DIALDroid-Bench
  - Real-world apps
  - No source code
  - No ground truth

```
import android.telephony.TelephonyManager;
/**
 * @testcase_name DirectLeak1
 * @version 0.1
 ...
 * @description Easy testcase: The value of a source is
      → directly sent to a sink
 * @dataflow source -> sink
 * @number_of_leaks 1
 * @challenges -
 */
public class MainActivity extends Activity {
```

Excerpt of the source code of one of DroidBench's app

Benchmarks

```
Found a flow to sink virtualinvoke $r4.<android.telephony.
    → SmsManager: void sendTextMessage(java.lang.String,
    → java.lang.String,java.lang.String,android.app.
    → PendingIntent,android.app.PendingIntent)>("+49 1234",
    →  null, $r5, null, null), from the following sources:
      - $r5 = virtualinvoke $r3.<android.telephony.
          → TelephonyManager: java.lang.String
          → getDeviceId()>() (in <de.ecspride.
          → MainActivity: void onCreate(android.os.Bundle
          → )>)
```

Listing 1: Excerpt of FlowDroid's result (DirectLeak1.apk)

```
### 'Sink: <android.telephony.SmsManager: void
    → sendTextMessage(java.lang.String,java.lang.String,
    → java.lang.String,android.app.PendingIntent,android.
    → app.PendingIntent)>': ###
['Src: <android.telephony.TelephonyManager: java.lang.String
    →  getDeviceId()>']
```

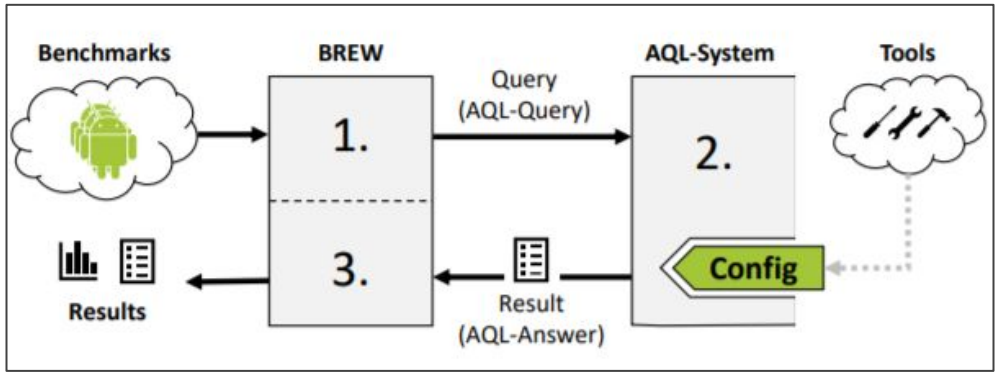
Listing 2: Excerpt of DidFail's result (DirectLeak1.apk)

Comparison of different analysis tools outputs.

Approach

ReproDroid - Overview

- A framework to support tools evaluation and comparison by following three concepts:
  - AQL (Android App Analysis Query Language)
  - AQL-System
  - Brew (Benchmark Refinement and Execution Wizard)



Sketch of ReproDroid toolchain

AQL (Android App Analysis Query Language)

- A query language for precisely formulating questions and answers about app properties such as flows.
- Two parts:
  - AQL-Queries
  - AQL-Answers

AQL-Queries

- AQL-Queries enables us to ask for Android specific analysis subjects in a general, tool independent way.
- AQL-Queries currently supports asking for analysis subjects such as flows, intents, intent-filters and permissions.

```
Flows IN App('/path/to/DirectLeak1.apk') ?
```

or instead explicitly check the taint flows we expect:

```
Flows FROM
    Statement('getDeviceId()')
->Method('onCreate(...)')->Class('MainActivity')
    ->App('/path/to/DirectLeak1.apk')
TO
    Statement('sendTextMessage(...)')
->Method('onCreate(...)')->Class('MainActivity')
    ->App('/path/to/DirectLeak1.apk')
?
```

AQL-Query example

AQL-Answers

- AQL-Answers enables us to represent analysis results standardized form.
- Syntax defined via an XML schema definition.

```
<answer>
    <flows>
        <flow>
            <reference type="from">
                <statement>... getDeviceId() ...</statement>
                <method>... onCreate(...) ...</method>
                <classname>... MainActivity</classname>
                <app>
                    <file>.../DirectLeak1.apk</file>
                    <hashes>...</hashes>
                </app>
            </reference>
            <reference type="to">
                ...
                sendTextMessage(...)
                ...
            </reference>
        </flow>
    </flows>
</answer>
```

AQL-Answer example

AQL-System

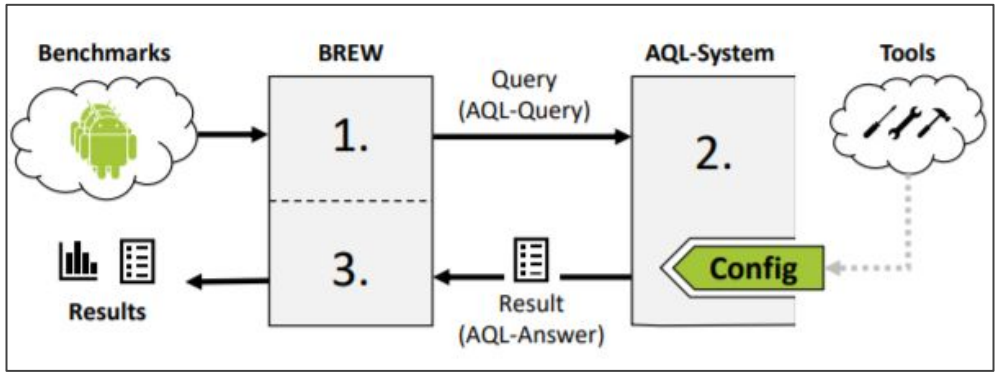
- Processes AQL-Queries and determines AQL-Answers
- XML configuration file
  - which tools are available and how to execute these
  - which queries can be answered by which tool
  - how to convert a tool's result into an AQL-Answer
- Given an AQL-Query:
  - launches the tools that may respond to it
  - translate the results into an AQL-Answer using the given tool-specific converters

Brew (Benchmark Refinement and Execution Wizard)

- An assistant to refine and execute benchmarks through a GUI.
- The proccess of refining benchmarks using Brew consists of three steps:
  - Case Identification
  - Source and Sink Identification
  - Ground Truth Identification
- Once the refinement steps have been completed, the benchmark can be executed and evaluated.
  - Brew determines one AQL-Query and one expected AQL-Answers per case
  - Brew sends a AQL-Query to each an AQL-System for each benchmark case and compares the actual result and expected result
  - Brew determines success and failures for each case and computes precision, recall and F-measure for the tools

ReproDroid

- Brew uses the AQL-System
- The AQL-System uses the AQL.
- Benchmarks and Analysis Tools exist in the community



Sketch of ReproDroid toolchain

Research Questions

RQ1. Do Android app analysis tools keep their promises?

- Determine what is a "promise"
  - The supported features
  - The tool's accuracy (precision, recall, F-measure)
  - Runtime appears to play a minor role
- Prepare a benchmark set (refined with Brew)
  - DroidBench
  - ICC-Bench
  - Their feature-checking benchmark cases
- Brew is launched six times
  - 1 AQL-System each time (1 tool at a time)
  - Default tool options

RQ2. How do the tools compare to each other with respect to accuracy?

- Choose F-measure as means for evaluating accuracy
  - For each category and tool the average value is computed
- Refined version of DroidBench
  - ICC-Bench is not used to avoid intermixing benchmark cases

RQ3. Which tools support large-scale analysis of real-world apps?

- Evaluate whether the tools are able to deal with
  - Large apps (in terms of code size)
  - A large numbers of apps
  - ICC and IAC
  - newer Android versions.
- Systematic derivation of a ground truth
  - Refined version of DIALDroid-Bench
- To check the tools ability for a large number of apps, ICC and IAC
  - Own intent-matching benchmarks
- To check applicability to new android APIs
  - Different versions of own feature-checking test apps and DroidBench

Setup

Tool Selection

- All tools selected implement ***taint*** analysis
- Only consider ***static*** taint analysis tools
- Consider only approaches that are at least flow-sensitive or context-sensitive

Tool Selection

- All six tools have at least ICC and at best IAC capabilities
- FlowDroid computes flows within single components only

Tool	Version
AMANDROID [30]	November 2017 (3.1.2)
DIALDROID [34]	September 2017
DIDFAIL [35]	March 2015
DROIDSAFE [36]	June 2016 (Final)
FLOWDROID [37]	April 2017* (Nightly)
IccTA [38]	February 2016

* Has been updated since then.

Tool Selection

- Analysis Engine
  - All tools except Amandroid are based on Soot and operate on Jimple as intermediate language
- Source and Sink Identification
  - Sources and sinks considered are specified by SuSi
  - For the micro-benchmarks, the sources and sinks needed for finding flows are identified by all tools
- ICC and IAC capabilities
  - Some tools are shipped with built-in IAC capabilities
  - Usage of ApkCombiner to take multiple .apk files as input and merge them into a single .apk file.

Tool	Version
AMANDROID [30]	November 2017 (3.1.2)
DIALDROID [34]	September 2017
DIDFAIL [35]	March 2015
DROIDSAFE [36]	June 2016 (Final)
FLOWDROID [37]	April 2017* (Nightly)
ICCTA [38]	February 2016

* Has been updated since then.

Benchmark Selection

- Experiments based on three benchmarks suites
  - Droid-Bench
  - ICC-Bench
  - DIALDroid-Bench
- 18 apps were developed comprising 21 positive and 6 negative ***feature-checking*** benchmark cases
  - Exploits only one specific feature at a time and can thus be used to explicitly check the handling of a dedicated feature in a tool.
- Three apps to specifically evaluate the precision of intent-matching algorithms were developed
- The analysis has to detect whether a certain intent can be received by a component. If so, the action, category and data attributes of an intent have to match those of a component's intent-filter

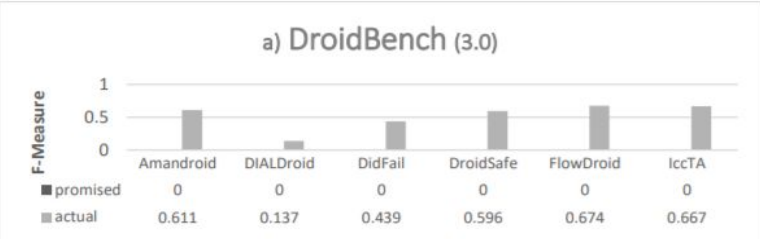
Results

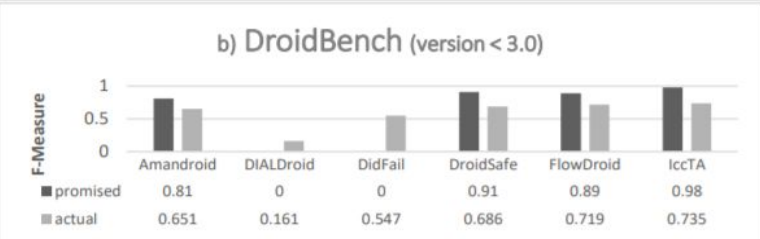
Feature Promises

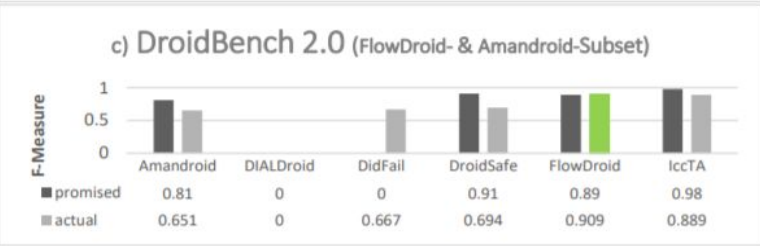
Tool	Aliasing	Static	Callbacks	Lifecycle	Inter-Procedural	Inter-Class	IAC	ICC (explicit)	ICC (implicit)	Sensitivity: Flow-	Sensitivity: Context-	Sensitivity: Field-	Sensitivity: Object-	Path-	ThreadAwareness	Reflection
AMANDROID	⊕	★	★	★	★	★	★	⊕	⊕	⊖	★	⊕	⊕	–	★	+
DIALDROID							⊖	★	⊖†							
DIDFAIL	○	○	○	○	○	○	○	○	○	○	○	○	○			
DROIDSAFE	⊕	★	⊖	★	★	★	⊕	⊕	⊕	–	⊖	⊕	⊕	–	★	+
FLOWDROID	⊕	★	★	★	★	★	–	–	–	★	★	⊕	★	–	★	–
ICCTA	⊕	★	★	★	★	★	★	★	⊕	★	★	⊕	★	–	★	–

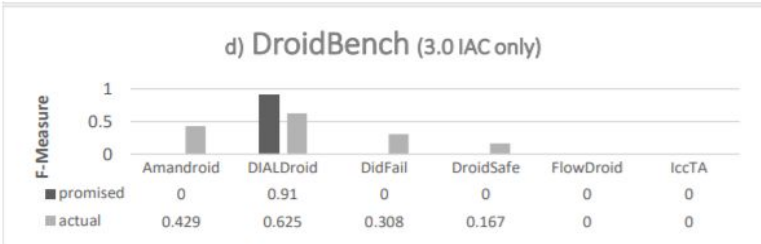
○ supported, ★ confirmed, + partially confirmed, – not confirmed, † aborted

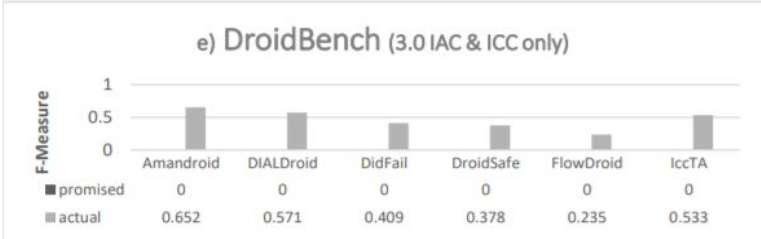
Accuracy Promises

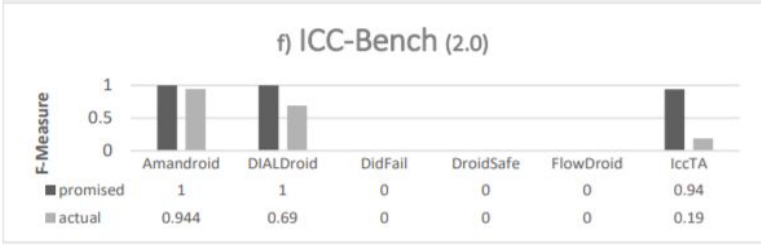










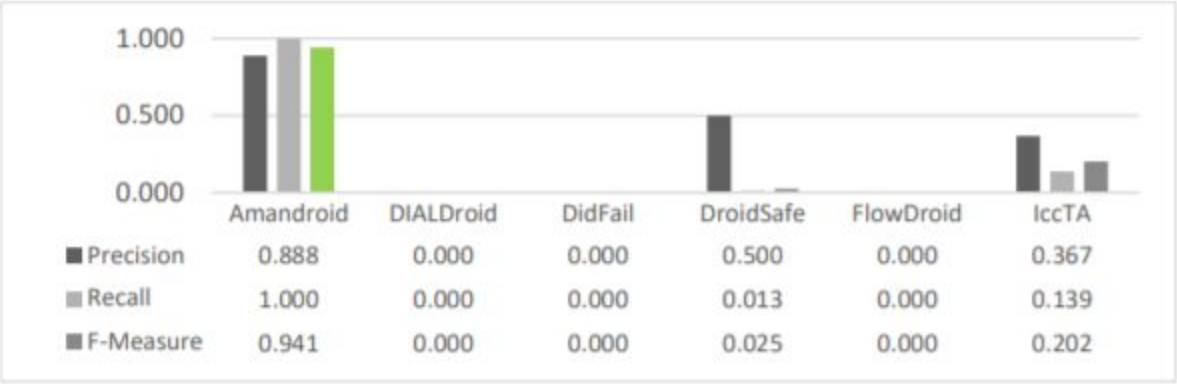


ID	Category	DIALDroid	DidFail	DroidSafe	IccTA	FlowDroid	Amandroid	Ø
1	FieldAndObjectSensitivity	0.000	0.800	0.667	1.000	1.000	1.000	0.745
2	Callbacks	0.000	0.769	0.667	0.897	0.897	0.500	0.622
3	UnreachableCode	0.000	1.000	0.000	0.857	1.000	0.857	0.619
4	AndroidSpecific	0.000	0.429	0.900	0.842	0.900	0.625	0.616
5	GeneralJava	0.000	0.611	0.780	0.762	0.810	0.703	0.611
6	EmulatorDetection	0.000	0.000	0.500	0.966	0.966	0.966	0.566
7	Lifecycle	0.000	0.400	0.933	0.737	0.769	0.545	0.564
8	InterComponentCommunication	0.538	0.452	0.480	0.706	0.348	0.750	0.546
9	Threading	0.000	0.667	0.000	0.667	1.000	0.667	0.500
10	ArraysAndLists	0.000	0.444	0.667	0.500	0.615	0.545	0.462
11	Aliasing	0.000	0.000	0.000	0.667	0.667	0.500	0.306
12	InterAppCommunication	0.625	0.308	0.167	0.000	0.000	0.429	0.255
13	Reflection	0.000	0.095	0.333	0.095	0.095	0.182	0.133
14	DynamicLoading	0.000	0.000	0.000	0.000	0.000	0.500	0.083
15	Native	0.000	0.000	0.000	0.000	0.000	0.333	0.056
16	ImplicitFlows	0.000	0.000	0.000	0.000	0.000	0.000	0.000
17	Reflection_ICC	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	SelfModification	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Ø	0.065	0.332	0.339	0.483	0.504	0.506	

F-Measure Scores

Tool	Number of successfully analyzed apps	Analysis-time per app (minutes)	Number of true / false positives
AMANDROID	21	8	1 / 0
DIALDROID	20	10	0 / 4
DIDFAIL	27	9	21 / 4
DROIDSAFE	2	5	0 / 0
FLOWDROID	18	2	22 / 0
IccTA	18	4	6 / 0

DIALDroid-Bench Results on real-world apps



	Amandroid	DIALDroid	DidFail	DroidSafe	FlowDroid	IccTA
Precision	0.888	0.000	0.000	0.500	0.000	0.367
Recall	1.000	0.000	0.000	0.013	0.000	0.139
F-Measure	0.941	0.000	0.000	0.025	0.000	0.202

Intent Matching: Precision, Recall, F-Measure

Tool	Eclipse	AndroidStudio	
	API ≤ 19	API 19	API 26
AMANDROID	✓	✓	–
DIALDROID	✓	✓	✓
DIDFAIL	✓	–†	–†
DROIDSAFE	✓	✓	–†
FLOWDROID	✓	✓	✓
IccTA	✓	✓	–†
APKCOMBINER	✓	✓	–†

✓ supported, – fails, † crashes

Up-to-date Status

Threats to Validity

- Manual (although tool-assisted) definition of the ground truth
  - No way around it, the tools that could derive the ground truth and the tools being evaluated are the same
  - Cannot rely on a single tool to generate the ground truth
  - ReproDroid allows us to refine the expected result definitions multiple times in the search for precision
- All tools were executed using their default configuration (except for available memory)
  - Some tools may reproduce different results using different parameters
  - A non-expert software developer os more likely to use those default settings
- Metrics precision
- AQL-System may contain bugs

Obrigado!

- Luca Ananias - lams3@cin.ufpe.br
- Andre De’ Carli - acms@cin.ufpe.br