Android Taint-Analysis Benchmarks: Past, Present and Future

Felix Pauck
7/17/2019
Android Taint-Analysis Benchmarks: Past, Present and Future

Overview

1. Overview
2. Competitions
3. Future
4. Competitions
5. Discussion
Taint Analysis

Source

Sink
Taint Analysis

Android
Taint-Analysis
Benchmarks:

Source

Sink

Static-Fields
Inter-Component
Path-Sensitivity
Thread-Awareness
IC
Intent
IC
Intent-Filter
Inter-Component
Object-Sensitivity
Field-Sensitivity
Lifecycle
Inter-App
Inter-Procedural

Aliasing
Callbacks
Manifest
Flow-Sensitivity
Reflection
Inter-Component
Context-Inter-Class

Present
Past
Future
Taint Analysis

App: A

1. S1: invoke(...) → s = getDeviceId()
   (reflective call)

2. S2: startActivity(...) → onCreate(...)

3. S3: getStringExtra() → onCreate(...) (Native library)

4. S4: Log.i(..., s)

5. S5: getIntent()

6. S6: cFuncSMS(s) (native call)

7. S7: sendTextMessage(..., s, ..)

8. S8: setResult(...)

App: B
Taint Analysis

App: A

- S1: invoke(...) → s = getDeviceId()
  *(reflective call)*
- S2: startActivity(...)
- S3: getStringExtra()
- S4: Log.i(..., s)

App: B

- S5: getIntent()
- S6: cFuncSMS(s) *(native call)*
- S7: sendTextMessage(.., s, ..)
- S8: setResult(...)

Native library
Taint Analysis

App: A

1. onCreate(..)
   - S1 invoke(...) → s = getDeviceId() (reflective call)
2. startActivity(..)

App: B

3. getIntent()

4. cFuncSMS(s) (native call)
   - Native library

5. sendTextMessage(., s, ..)

6. onActivityResult(..)
   - S3 getStringExtra()

7. log.i(., s)

8. setResult(..)
Android Taint-Analysis Benchmarks:

Taint Analysis

App: A

1. s = getDeviceId()
   (reflective call)

2. startActivity(...)

3. getIntent()

4. cFuncSMS(s)
   (native call)

5. Native library

6. sendTextMessage(..., s, ..)

7. Log.i(..., s)

8. setResult(...)

App: B

onCreate(...)

onCreate(...)

onStart(...)

onStart(...)

onStart(...)

onStart(...)

onStart(...)

onStart(...)

onStart(...)
Taint Analysis

App: A

S1

invoke(...) →

S = getDeviceId()

(Reflective call)

S2

startActivity(...)  

S3

getStringExtra()  

S4

Log.i(..., S)

App: B

S5

getIntent()  

S6

cFuncSMS(S)

(native call)

S7

,! Native library

sendTextMessage(..., S, ..)

S8

setResult(...)
Taint Analysis

App: A

1. `invoke(...) \rightarrow s = getDeviceId()` (reflective call)

2. `startActivity(...)`

App: B

3. `getIntent()`

4. `cFuncSMS(s)` (native call)

5. `onCreate(...)`

6. `getStringExtra()`

7. `onActivityResult(...)`

8. `setResult(...)`

9. `Log.i(..., s)`

Native library

`sendTextMessage(..., s, ...)"
Taint Analysis

Tools

- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- ...

Android Taint-Analysis Benchmarks:

Source

Sink

Tools

- Static-Fields
- Inter-Component
- Path-Sensitivity
- Aliasing
- Thread-Awareness
- Callbacks
- Intent-IAC
- Manifest
- Flow-Sensitivity
- Reflection
- Intetnt-Filter
- Lifecycle
- Inter-App
- Inter-Component
- Inter-Procedural
Android Taint-Analysis Benchmarks:

- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- ...

Tools:

- Static-Fields
- Inter-Component
- Path-Sensitivity
- Thread-Awareness
- Intent IAC
- Context-Sensitivity
- Aliasing
- Manifest
- Flow-Sensitivity
- Intent-Filter
- Reflection
- Lifecycle
- Inter-App
- Inter-Procedural

Taint Analysis
Android Taint-Analysis Benchmarks:

- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- ...

Tools

Benchmarks

- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- ...

Taint Analysis

Source

Sink
Android Taint-Analysis Benchmarks:

- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …

Tools:
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- …

Benchmarks:

- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …

Actual results:
- Static-Fields
- Inter-Component
- Path-Sensitivity
- Aliasing
- Callbacks
- Intent
- IAC
- Manifest
- Flow-Sensitivity
- Intent-Filter
- Reflection
- Lifecycle
- Inter-App
- Inter-Class
- Inter-Procedural
- …
**Tools**
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA

**Sources**

**Tools**
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- ...

**Ground-truth / Expected results**

**Actual results**

**Benchmarks**

**Taint Analysis**
Android Taint-Analysis Benchmarks:

- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- ...
Android Taint-Analysis Benchmarks:

Taint Analysis

Source → Sink

Tools:
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA

Benchmarks:
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- ...

Actual results ≠
Ground-truth / Expected results
Benchmarks: Past
Benchmarks: Past


B. RQ2: Experimental Results on Real-World Apps

To evaluate our approach, we launch IccTA on two Android

performs a simple string analysis to distinguish the extra keys of an Intent between one another.

IccTA outperforms both the commercial and academic tools by achieving a precision of 96.6% and a recall of 96.6% on DroidBench and ICC-Bench.
Standardized evaluation procedure,

\[ F\text{-Measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]
+ Standardized evaluation procedure,
+ Large userbase,
Standardized evaluation procedure,
Large userbase,
A collection of deprecated apps,
Benchmarks: Past

- Standardized evaluation procedure,
- Large userbase,
- A collection of deprecated apps,
- Not executable,
- Manual evaluation,
Standardized evaluation procedure,

- Large userbase,
- A collection of deprecated apps,
- Not executable,
- Manual evaluation,
- No reproducibility.

```java
import android.widget.Button;

/**
 * @testcase_name StartActivityForResult
 * @version 0.1
 * @author Contributed by [Contributor]
 * @author_mail (Maintainer) [Maintainer]
 *
 * @description Reads the user's geographical location (via GPS) and leaks it to the file system, and passes it to another activity using startActivityForResult which writes it to a file.
 * @dataflow getLastKnownLocation -> startActivityForResult -> onActivityResult -> FileOutputStream
 * @number_of_leaks 1
 * @challenges Inter-component communication using startActivityForResult must be handled correctly
 */

public class MainActivity extends Activity {
```
Benchmarks: Past

+ Standardized evaluation procedure,
+ Large userbase,
+ A collection of deprecated apps,
+ Not executable,
+ Manual evaluation,
+ No reproducibility.

“The term benchmark is utilized for the benchmarking programs themselves”*
Standardized evaluation procedure,
Large userbase,
A collection of deprecated apps,
Not executable,
Manual evaluation,
No reproducibility.

“The term benchmark is utilized for the benchmarking programs themselves”*

“a benchmark is the act of running a program in order to assess the relative performance”*

* shortened
https://en.wikipedia.org/wiki/Precision_and_recall
https://github.com/secure-software-engineering/DroidBench/tree/develop
https://en.wikipedia.org/wiki/Benchmark_(computing)
Benchmarks: Present
Do Android Taint Analysis tools keep their promises? [ESEC/FSE’18]
Do Android Taint Analysis tools keep their promises? [ESEC/FSE’18]
Benchmarks: Present

Do Android Taint Analysis tools keep their promises? [ESEC/FSE’18]

1. Refine: Colloquial ground-truth → Machine readable (semi-automatic)
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
Do Android Taint Analysis tools keep their promises? [ESEC/FSE’18]

1. **Refine**: Colloquial ground-truth → Machine readable
   (semi-automatic)
2. **Execute**: Run arbitrary tools
   (automatic)
3. **Collect & Evaluate**: Precision, Recall, F-measure
   (automatic)
Benchmarks: Present

Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]

1. **Refine**: Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute**: Run arbitrary tools (automatic)
3. **Collect & Evaluate**: Precision, Recall, F-measure (automatic)

**Benchmarks**

**Tools**

**Easy to add new/more tools:**
1. Adjust config
2. Add converter (if needed)

**Config**

**Benchmarking Framework:**
reproDroid

Android Taint-Analysis Benchmarks:

Present Past Future
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]

1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]

**Benchmarks: Present**

Flows FROM App('A.apk') TO App('B.apk')?

MATCH [
  Flows IN App('A.apk' | 'DEOBFUSCATE') ?,
  CONNECT [
    Flows IN App('B.apk' | 'UNCOVER') ?,
    Flows IN App('B.apk' | 'UNCOVER') FEATURING 'NATIVE' ?
  ],
  IntentSources IN App('A.apk' | 'DEOBFUSCATE') ?,
  IntentSinks IN App('A.apk' | 'DEOBFUSCATE') ?,
  IntentSources IN App('B.apk' | 'UNCOVER') ?,
  IntentSinks IN App('B.apk' | 'UNCOVER') ?
]

1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]

1. **Refine:** Colloquial ground-truth $\rightarrow$ Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)
1. **Refine**: Colloquial ground-truth → Machine readable  
   (semi-automatic)

2. **Execute**: Run arbitrary tools  
   (automatic)

3. **Collect & Evaluate**: Precision, Recall, F-measure  
   (automatic)
+ Standardized evaluation procedure,
+ Large userbase,
- A collection of deprecated apps,
- Not executable,
- Manual evaluation,
- No reproducibility.
Standardized evaluation procedure,
Large userbase,
A collection of deprecated apps,
Executable,
Automatic evaluation,
Reproducible.
Benchmarks: Present

+ Standardized evaluation procedure,
+ Large userbase,
- A collection of deprecated apps,
+ Executable,
+ Automatic evaluation,
+ Reproducible.

“a benchmark is the act of running a program in order to assess the relative performance”*
Benchmarks: Present

- Standardized evaluation procedure,
- Large userbase,
- A collection of deprecated apps,
- Executable,
- Automatic evaluation,
- Reproducible.

- Exemplary Use-Cases:
  - Evaluate novel tools
  - Continuous Integration
  - ...

```
"a benchmark is the act of running a program in order to assess the relative performance"
```
• Evaluate novel tools

Together Strong: Cooperative Android App Analysis [ESEC/FSE'19]
Evaluate novel tools

MATCH [
  Flows IN App('A.apk' | 'DEOBFSUicate') ?,
  CONNECT [
    Flows IN App('B.apk' | 'UNCOVER') ?,
    Flows IN App('B.apk' | 'UNCOVER') FEATURING 'NATIVE' ?
  ],
  IntentSources IN App('A.apk' | 'DEOBFSUicate') ?,
  IntentSinks IN App('A.apk' | 'DEOBFSUicate') ?,
  IntentSources IN App('B.apk' | 'UNCOVER') ?,
  IntentSinks IN App('B.apk' | 'UNCOVER') ?
]
### android Taint-Analysis Benchmarks

#### Present

Together Strong: Cooperative Android App Analysis [ESEC/FSE’19]

- Evaluate novel tools

#### Table: Benchmarks

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>FlowDroid</th>
<th>Best</th>
<th>CoDrDroid</th>
<th>Difference to Best</th>
<th>Difference to FlowDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aliasing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \emptyset \)
## Android Taint-Analysis Benchmarks: Present

**Benchmarks:** Present

### Together Strong: Cooperative Android App Analysis [ESEC/FSE'19]

#### Evaluate novel tools

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>FlowDroid</th>
<th>Best</th>
<th>CoDroid</th>
<th>Difference to Best</th>
<th>Difference to FlowDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aliasing</td>
<td>0.667</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td>0.900</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td>0.615</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td>0.897</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td>0.966</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td>0.810</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td>0.348</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td>0.769</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td>0.095</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ø</td>
<td></td>
<td>0.504</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Android Taint-Analysis Benchmarks

**Benchmarks:** Present

**Together Strong: Cooperative Android App Analysis [ESEC/FSE’19]**

- Evaluate novel tools
- Config: Together Strong: Cooperative Android App Analysis

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>FlowDroid</th>
<th>Best</th>
<th>CoDiDroid</th>
<th>Difference to Best</th>
<th>Difference to FlowDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aliasing</td>
<td>0.667</td>
<td>0.667</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td>0.900</td>
<td>0.900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td>0.615</td>
<td>0.615</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td>0.897</td>
<td>0.897</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td>0.966</td>
<td>0.966</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td>1.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td>0.810</td>
<td>0.810</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td>0.000</td>
<td>0.625</td>
<td>0.625</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td>0.348</td>
<td>0.690</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td>0.769</td>
<td>0.769</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td>0.000</td>
<td>0.889</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td>0.095</td>
<td>0.800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td>1.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td>1.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ø</td>
<td></td>
<td>0.504</td>
<td>0.646</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Android Taint-Analysis Benchmarks

**Present**

**Benchmarks: Present**

Together Strong: Cooperative Android App Analysis [ESEC/FSE’19]

- Evaluate novel tools

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>FlowDroid</th>
<th>Best</th>
<th>CoDiDroid</th>
<th>Difference to Best</th>
<th>Difference to FlowDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aliasing</td>
<td>0.667</td>
<td></td>
<td>0.667</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td>0.900</td>
<td></td>
<td>0.900</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td>0.615</td>
<td></td>
<td>0.615</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td>0.897</td>
<td></td>
<td>0.897</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td>0.000</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td>0.966</td>
<td></td>
<td>0.966</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td>1.000</td>
<td></td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td>0.810</td>
<td></td>
<td>0.810</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td>0.000</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td>0.000</td>
<td></td>
<td>0.625</td>
<td>-0.625</td>
<td>-0.625</td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td>0.348</td>
<td></td>
<td>0.690</td>
<td>-0.342</td>
<td>-0.342</td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td>0.769</td>
<td></td>
<td>0.769</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td>0.000</td>
<td></td>
<td>0.889</td>
<td>-0.889</td>
<td>-0.889</td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td>0.095</td>
<td></td>
<td>0.800</td>
<td>-0.705</td>
<td>-0.705</td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td>0.000</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td>0.000</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td>1.000</td>
<td></td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td>1.000</td>
<td></td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Ø</td>
<td></td>
<td>0.504</td>
<td></td>
<td>0.646</td>
<td>-0.142</td>
<td>-0.142</td>
</tr>
<tr>
<td>ID</td>
<td>Category</td>
<td>FlowDroid</td>
<td>Best</td>
<td>CoDiDroid</td>
<td>Difference to Best</td>
<td>Difference to FlowDroid</td>
</tr>
<tr>
<td>----</td>
<td>------------------------------</td>
<td>-----------</td>
<td>------</td>
<td>-----------</td>
<td>--------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Aliasing</td>
<td>0.667</td>
<td>0.667</td>
<td>0.667</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td>0.900</td>
<td>0.900</td>
<td>0.900</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td>0.615</td>
<td>0.667</td>
<td>0.615</td>
<td>0.052</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td>0.897</td>
<td>0.897</td>
<td>0.897</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td>0.000</td>
<td>0.500</td>
<td>0.000</td>
<td>0.500</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td>0.966</td>
<td>0.966</td>
<td>0.966</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td>0.810</td>
<td>0.810</td>
<td>0.810</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td>0.000</td>
<td>0.625</td>
<td>0.625</td>
<td>0.000</td>
<td>-0.625</td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td>0.348</td>
<td>0.750</td>
<td>0.690</td>
<td>0.060</td>
<td>-0.342</td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td>0.769</td>
<td>0.933</td>
<td>0.769</td>
<td>0.164</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td>0.000</td>
<td>0.333</td>
<td>0.889</td>
<td>-0.556</td>
<td>-0.889</td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td>0.095</td>
<td>0.333</td>
<td>0.800</td>
<td>-0.467</td>
<td>-0.705</td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Ø</td>
<td></td>
<td>0.504</td>
<td>0.632</td>
<td>0.646</td>
<td>-0.014</td>
<td>-0.142</td>
</tr>
</tbody>
</table>
• Continuous Integration
  o Requires: Benchmark optimization
  o Approach: App merging
• Continuous Integration
  o Requires: Benchmark optimization
  o Approach: App merging
• Continuous Integration
  o Requires: Benchmark optimization
  o Approach: App merging
Continuous Integration

![Graph showing execution time for one by one and merged approaches with Droidbench Category Size (#number of apps)](image-url)
• Continuous Integration
  o Requires: Benchmark optimization
  o Approach: App merging
Android Taint-Analysis Benchmarks:

Benchmarks: Future

Future
Standardized evaluation procedure,
Large userbase,
A collection of deprecated apps,
Executable,
Automatic evaluation,
Reproducible.
+ Standardized evaluation procedure,
+ Large userbase,
- A collection of deprecated apps,
+ Executable,
+ Automatic evaluation,
+ Reproducible.
+ Standardized evaluation procedure,
+ Large userbase,
+ A collection of up-to-date apps,
+ Executable,
+ Automatic evaluation,
+ Reproducible.

**ISSTA 2019 Technical Papers**

**Accepted Papers**

- Title

  - A Large-Scale Study of Application Incompatibilities in Android
    Haipeng Cai, Ziyi Zhang, Li Li, Xiaojin Fu
    Pre-print
  - Adlib: Analyzer for Mobile Ad Platform Libraries
    Sungho Lee, Sukyoung Ryu
  - Assessing the State and Improving the Art of Parallel Testing for C:
    Alessio Gambi, Marc Mueller, Gordon Fraser

**Automated API-Usage Update for Android Apps**

Mattia Fazzini, Qi Xin, Alessandro Orso
Standardized evaluation procedure,
Large userbase,
A collection of up-to-date apps,
Executable,
Automatic evaluation,
Reproducible.

Competitions

Automated API-Usage Update for Android Apps
Mattia Fazzini, Qi Xin, Alessandro Orso
Android Taint-Analysis

Benchmarks: Future

Competition Diagram:
- SV-COMP
- SMT-COMP
- MCC
- REC
- CRV
- SL-COMP
- VerifyThis
- Test-Comp
- MCC
- Termination Competition
- COCO
• Challenges

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Author</th>
<th>App (.apk)</th>
<th>Source code (.zip)</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Native</td>
<td>foellix.github.io</td>
<td>SourceInNative.apk</td>
<td>SourceInNative.zip</td>
<td>0_tp.xml</td>
<td>0_fp.xml</td>
</tr>
</tbody>
</table>

• Participants
  1. Analysis tool

• Committee
  1. Review:
     o Challenge submissions
     o Tool submissions (+ associated tool-papers)
  2. Run the competition
• **Challenges**

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Author</th>
<th>App (.apk)</th>
<th>Source code (.zip)</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Native</td>
<td>foellix.github.io</td>
<td>SourceInNative.apk</td>
<td>SourceInNative.zip</td>
<td>0 tp.xml</td>
<td>0 fp.xml</td>
</tr>
</tbody>
</table>

• **Participants**
  1. Analysis tool

• **Committee**
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition
Benchmarks: Future

- Challenges

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Author</th>
<th>App (.apk)</th>
<th>Source code (.zip)</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Native</td>
<td>foellix.github.io</td>
<td>SourceInNative.apk</td>
<td>SourceInNative.zip</td>
<td>0Tp.xml</td>
<td>0_fp.xml</td>
</tr>
</tbody>
</table>

- Participants
  1. Analysis tool

- Committee
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition
Benchmarks: Future

- **Challenges**
  - ID | Category | Author         | App (.apk)        | Source code (.zip) | True positives | False positives |
  - 0  | Native   | foellix.github.io | SourceInNative.apk | SourceInNative.zip | 0_tp.xml       | 0_fp.xml       |

- **Participants**
  1. Analysis tool

- **Committee**
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition
Benchmarks: Future

- Challenges
<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Author</th>
<th>App (.apk)</th>
<th>Source code (.zip)</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Native</td>
<td>foellix.github.io</td>
<td>SourceInNative.apk</td>
<td>SourceInNative.zip</td>
<td>0_tp.xml</td>
<td>0_fp.xml</td>
</tr>
</tbody>
</table>

- Participants
  1. Analysis tool
  2. Configuration
  3. Converter

- Committee
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition
• Challenges

<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>Author</th>
<th>App (.apk)</th>
<th>Source code (.zip)</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Native</td>
<td>foellix.github.io</td>
<td>SourceInNative.apk</td>
<td>SourceInNative.zip</td>
<td>0_tp.xml</td>
<td>0_fp.xml</td>
</tr>
</tbody>
</table>

• Participants

1. Analysis tool
2. **Configuration**
3. **Converter**

• Committee

1. Review:
   1. Challenge submissions
   2. Tool submissions (+ associated tool-papers)
2. Run the competition

When do we start an Android Taint-Analysis Competition?
**Android Taint-Analysis Benchmarks:**

**Benchmarks: Future**

- **Challenges**
  - ID
  - Category
  - Author
  - False positives

**ASAP**, we are ready and it boosts tool and benchmark development!

**Organization, Venue, Prices, Proceedings (Competition report), ... ?**

**Committee**

1. Review:
   - Challenge submissions
   - Tool submissions (+ associated tool-papers)
2. Run the competition

**When do we start an Android Taint-Analysis Competition?**
Android Taint-Analysis Benchmarks: Got, Get and will get better!

**Summary**

**Past 😞**
- Eva. procedure,
- Userbase,
- Up-to-date,
- Executability,
- Automation,
- Reproducibility.

Github: [https://FoelliX.github.io](https://FoelliX.github.io)
**Android Taint-Analysis Benchmarks:** Got, Get and will get better!

**Summary**

- **Past**
  - Eval. procedure,
  - Userbase,
  - Up-to-date,
  - Executability,
  - Automation,
  - Reproducibility.

- **Present**
  - Executability,
  - Automation,
  - Reproducibility.

**Github:** https://FoelliX.github.io
Android Taint-Analysis Benchmarks: Got, Get and will get better!

**Past 😞**  
+ Eva. procedure,  
+ Userbase,  
- Up-to-date,  
- Executability,  
- Automation,  
- Reproducibility.

**Present 😊**  
- Executability,  
- Automation,  
- Reproducibility.

**Future 😊**  
+ Up-to-date,  
+ Executability,  
+ Automation,  
+ Reproducibility.

Github: [https://FoelliX.github.io](https://FoelliX.github.io)
### Android Taint-Analysis Benchmarks: Got, Get and will get better!

<table>
<thead>
<tr>
<th>Past</th>
<th>Present</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>🙁</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>+ Eva. procedure,</td>
<td>+ Executability,</td>
<td>+ Up-to-date,</td>
</tr>
<tr>
<td>+ Userbase,</td>
<td>+ Automation,</td>
<td></td>
</tr>
<tr>
<td>- Up-to-date,</td>
<td>- Executability,</td>
<td>+ Executability,</td>
</tr>
<tr>
<td>- Executability,</td>
<td>- Automation,</td>
<td>+ Automation,</td>
</tr>
</tbody>
</table>

Github: [https://FoelliX.github.io](https://FoelliX.github.io)
Example (1/3)
Example (2/3)

Flows FROM App('A.apk') TO App('B.apk') ?

MATCH [
  Flows IN App('A.apk' | 'DEOBFUSCATE') ?,
  CONNECT [
    Flows IN App('B.apk' | 'UNCOVER') ?,
    Flows IN App('B.apk' | 'UNCOVER') FEATURING 'NATIVE' ?
  ],
  IntentSources IN App('A.apk' | 'DEOBFUSCATE') ?,
  IntentSinks IN App('A.apk' | 'DEOBFUSCATE') ?,
  IntentSources IN App('B.apk' | 'UNCOVER') ?,
  IntentSinks IN App('B.apk' | 'UNCOVER') ?
]
Example (3/3)

-Answer

```
[de.foellix.aql.codidroid.codirunex.SMS] []
[de.foellix.aql.codidroid.codirunex.SMS] [android.intent.category.DEFAULT]
```

```
getDeviceId()
getEmail()
.MainActivity
AppA_preprocessed-1.apk

getActivityForResult()() onActivityCreated()
.onCreate(android.os.Bundle)
.MainActivity
AppA_preprocessed-1.apk

startActivityForResult()()
.onCreate(android.os.Bundle)
.MainActivity
AppA_preprocessed-1.apk

getIntent()
.onCreate(android.os.Bundle)
.MainActivity
AppA_preprocessed-1.apk

cFuncSMS(java.lang.String)
sink(java.lang.String)
.MainActivity
AppB_preprocessed-1.apk

sendTextMessage()
.sink(java.lang.String)
.MainActivity
AppB.apk
```
1. **Refine**: Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute**: Run arbitrary tools (automatic)
3. **Collect & Evaluate**: Precision, Recall, F-measure (automatic)

Easy to add new/more tools:
1. Adjust config
2. Add converter (if needed)
<table>
<thead>
<tr>
<th>ID</th>
<th>Category</th>
<th>FlowDroid</th>
<th>Best</th>
<th>CoDiDroid</th>
<th>Difference to Best</th>
<th>Difference to FlowDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aliasing</td>
<td>0.667</td>
<td>0.667</td>
<td>0.667</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>AndroidSpecific</td>
<td>0.900</td>
<td>0.900</td>
<td>0.900</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>ArraysAndLists</td>
<td>0.615</td>
<td>0.667</td>
<td>0.615</td>
<td>0.052</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>Callbacks</td>
<td>0.897</td>
<td>0.897</td>
<td>0.897</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>DynamicLoading</td>
<td>0.000</td>
<td>0.500</td>
<td>0.000</td>
<td>0.500</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>EmulatorDetection</td>
<td>0.966</td>
<td>0.966</td>
<td>0.966</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>7</td>
<td>FieldAndObjectSensitivity</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>GeneralJava</td>
<td>0.810</td>
<td>0.810</td>
<td>0.810</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>ImplicitFlows</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>InterAppCommunication</td>
<td>0.000</td>
<td>0.625</td>
<td>0.625</td>
<td>0.000</td>
<td>-0.625</td>
</tr>
<tr>
<td>11</td>
<td>InterComponentCommunication</td>
<td>0.348</td>
<td>0.750</td>
<td>0.690</td>
<td>0.060</td>
<td>-0.342</td>
</tr>
<tr>
<td>12</td>
<td>Lifecycle</td>
<td>0.769</td>
<td>0.933</td>
<td>0.769</td>
<td>0.164</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>Native</td>
<td>0.000</td>
<td>0.333</td>
<td>0.889</td>
<td>-0.556</td>
<td>-0.889</td>
</tr>
<tr>
<td>14</td>
<td>Reflection</td>
<td>0.095</td>
<td>0.333</td>
<td>0.800</td>
<td>-0.467</td>
<td>-0.705</td>
</tr>
<tr>
<td>15</td>
<td>Reflection_ICC</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>SelfModification</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>Threading</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>UnreachableCode</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Ø</td>
<td></td>
<td>0.504</td>
<td>0.632</td>
<td>0.646</td>
<td>-0.014</td>
<td>-0.142</td>
</tr>
</tbody>
</table>
Android Taint-Analysis Benchmarks:

AMT

Input Apps (.apk)

Phase-I

1. Parse Classes
2. Parse Manifests
3. Resolve Conflicts
4. Build dummy lifecycle functions
5. Merge

Phase-II

6. Analyze Apps
7. Compare & Evaluate

[optional]

Merged App (.apk)

Report

reproDroid

Android Benchmark Reproduction Framework
Flows IN App('/path/to/DirectLeak1.apk')

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')
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
→ ))

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

```xml
<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>
```
• Android Taint-Analysis Benchmarks:

AQL: Add a new/more tools

- Analysis tools, Preprocessors, Operators
- Converter

• ~ 140 LOC for our 6 converters
Android Taint-Analysis Benchmarks: