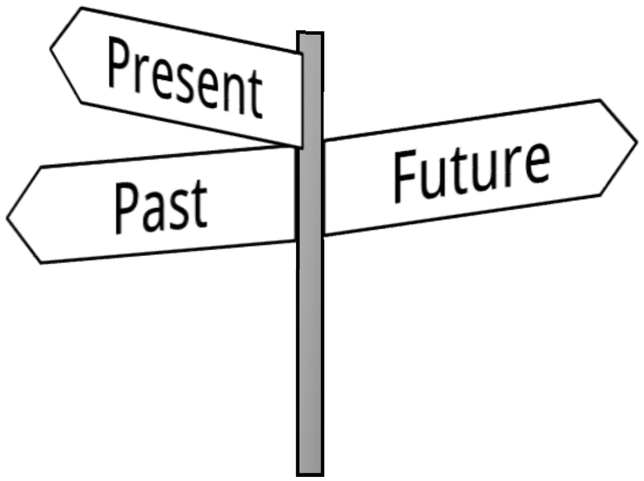
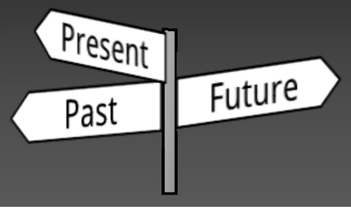
Android Taint-Analysis Benchmarks: Past, Present and Future



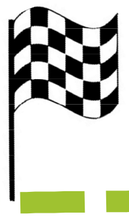
Felix Pauck

7/17/2019

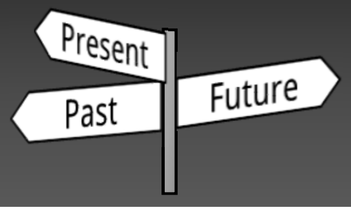


Android Taint-Analysis Benchmarks: Past, Present and Future

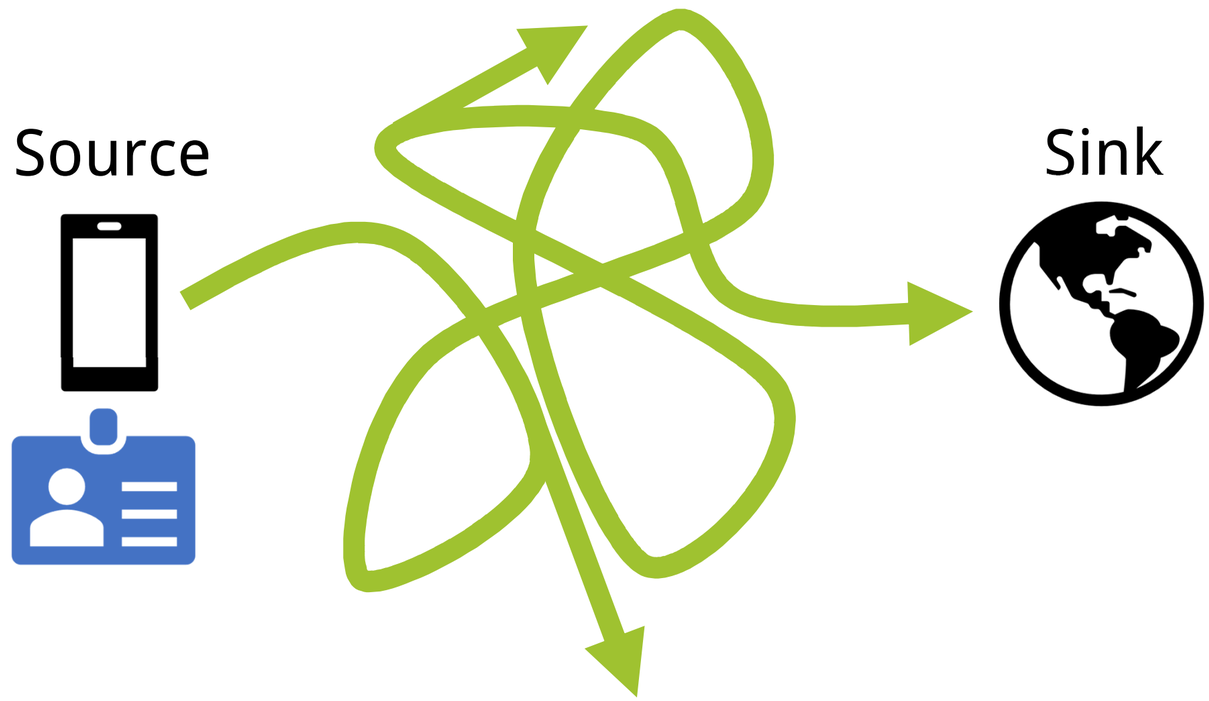
1. Taint-Analysis
2. 
3. 

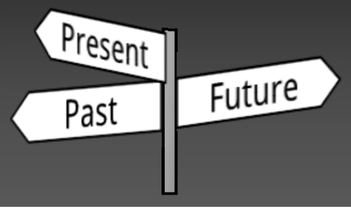
4. Competitions
5. 

Discussion

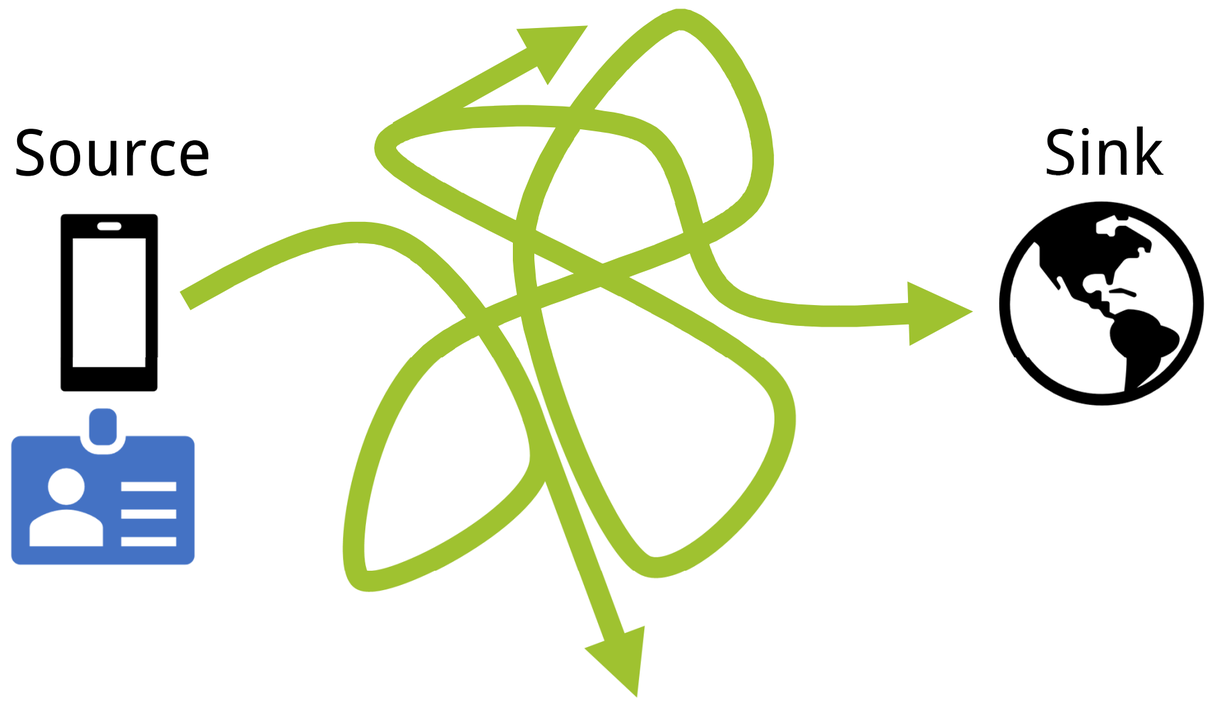


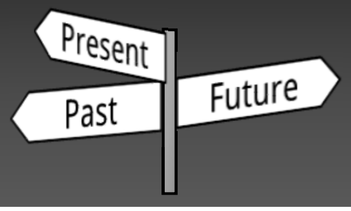
Taint Analysis



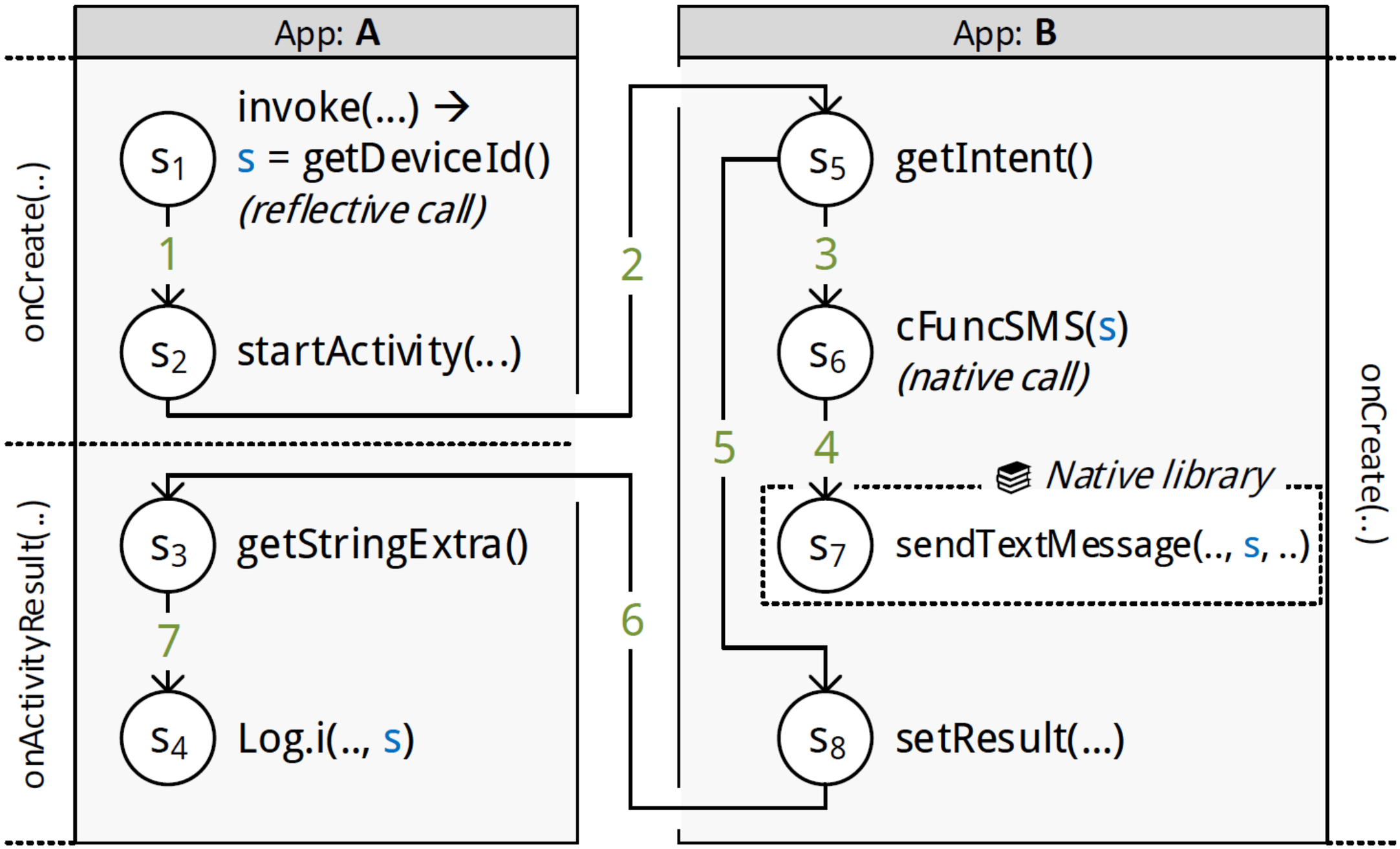


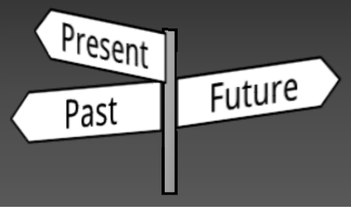
Taint Analysis



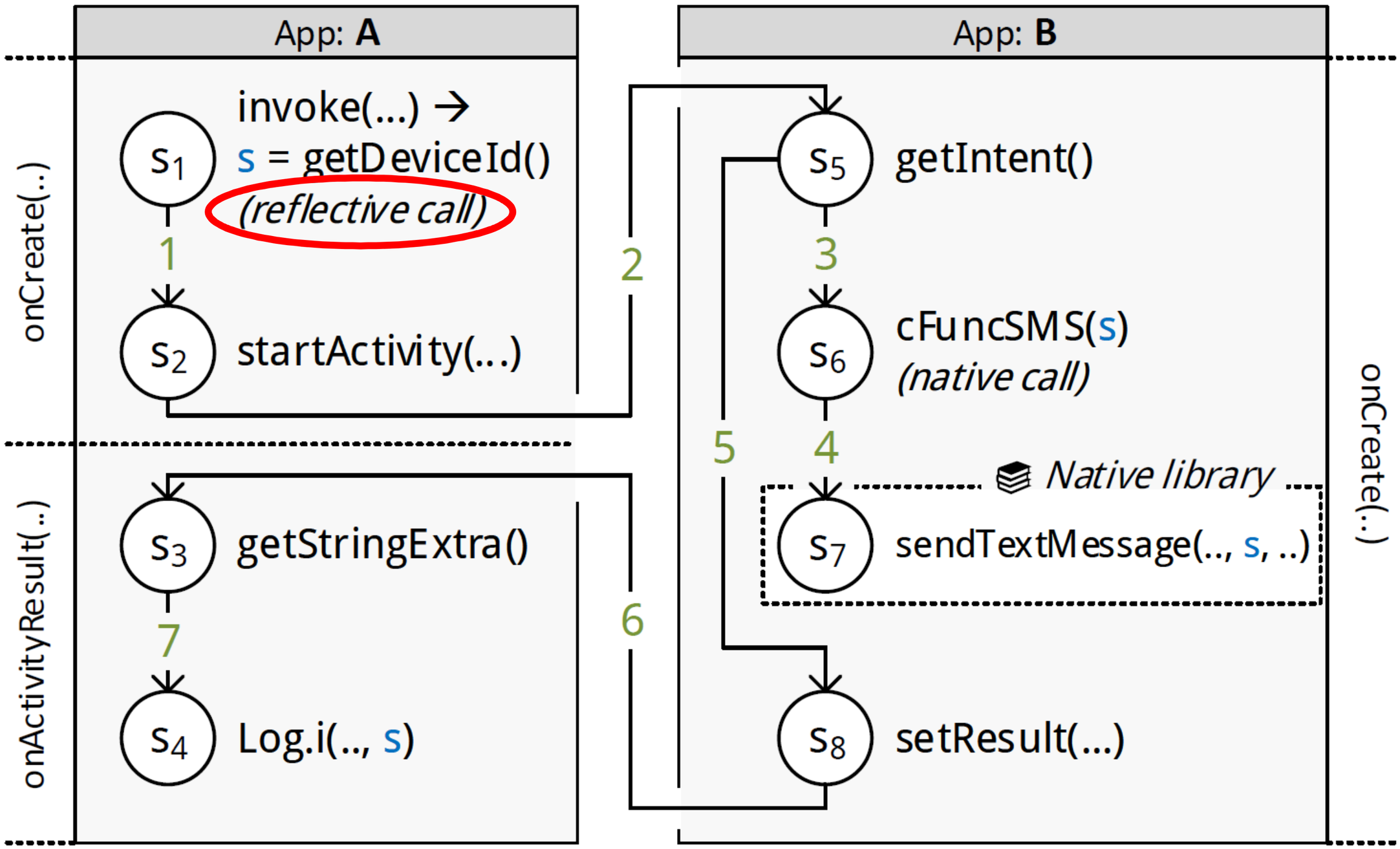


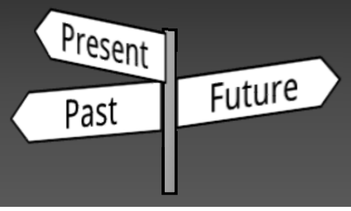
Taint Analysis



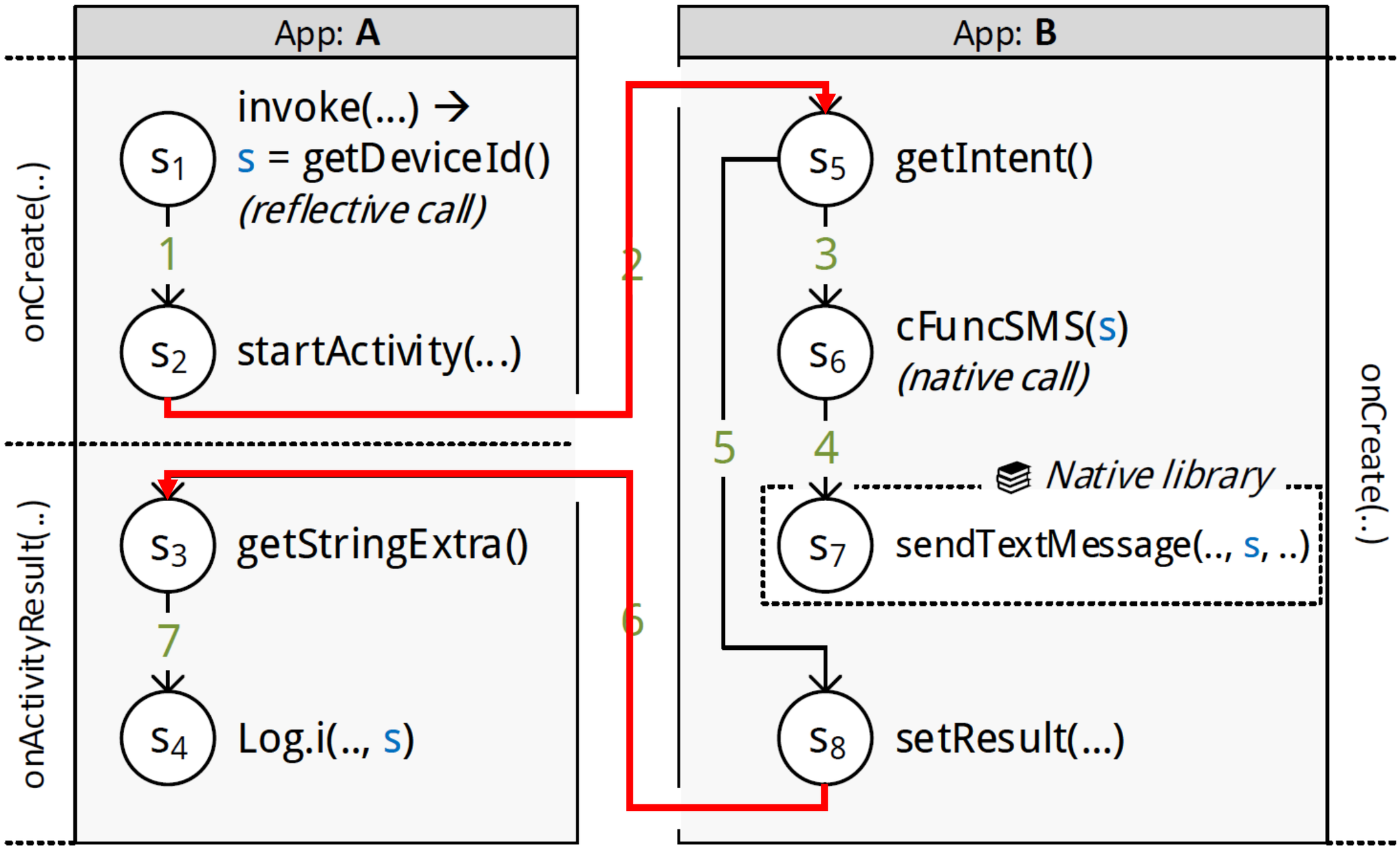


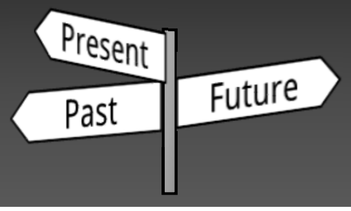
Taint Analysis





Taint Analysis





Taint Analysis

App: **A**

onCreate(..)

s_1 invoke(...) → s = getDeviceId() *(reflective call)*

1

s_2 startActivity(...)

onActivityResult(..)

s_3 getStringExtra()

7

s_4 Log.i(.., s)

2

6

App: **B**

onCreate(..)

s_5 getIntent()

3

s_6 cFuncSMS(s) *(native call)*

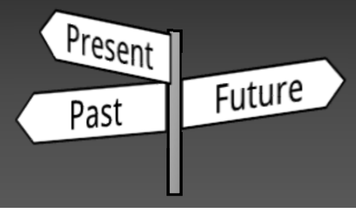
5

4

Native library

s_7 sendTextMessage(.., s, ..)

s_8 setResult(...)



Taint Analysis

App: **A**

App: **B**

onCreate(..)

onActivityResult(..)

onCreate(..)

s_1 invoke(...) →
s = getDeviceId()
(reflective call)

s_2 startActivity(...)

s_3 getStringExtra()

s_4 Log.i(.., s)

s_5 getIntent()

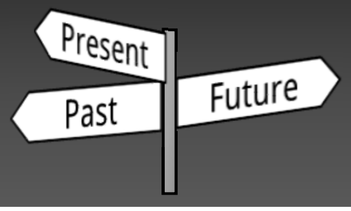
s_6 cFuncSMS(s)
(native call)

Native library

s_7 sendTextMessage(.., s, ..)

s_8 setResult(...)

1 2 3 4 5 6 7



Taint Analysis

App: **A**

App: **B**

onCreate(..)

onActivityResult(..)

onCreate(..)

s_1 invoke(...) → s = getDeviceId() *(reflective call)*

s_2 startActivity(...)

s_3 getStringExtra()

s_4 Log.i(.., s)

s_5 getIntent()

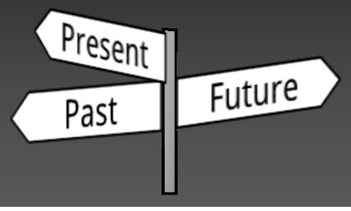
s_6 cFuncSMS(s) *(native call)*

Native library

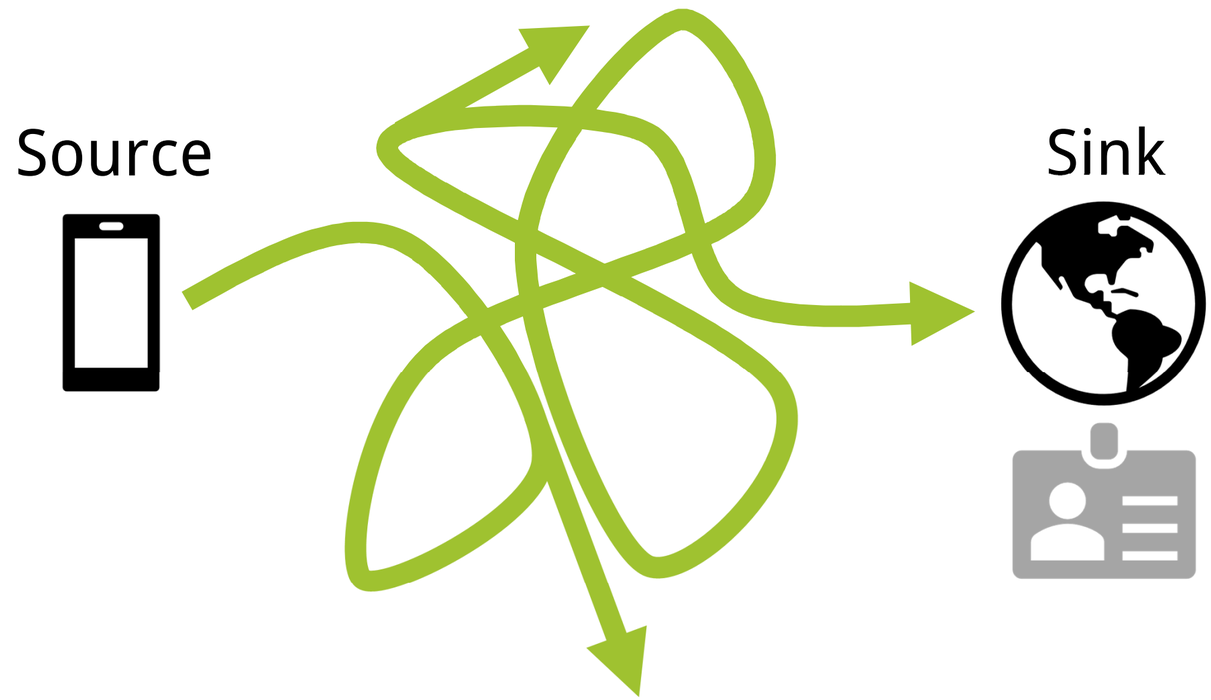
s_7 sendTextMessage(.., s, ..)

s_8 setResult(...)

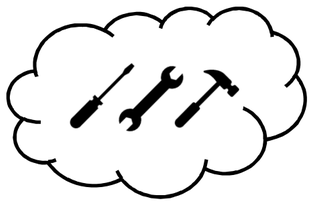
1 2 3 4 5 6 7



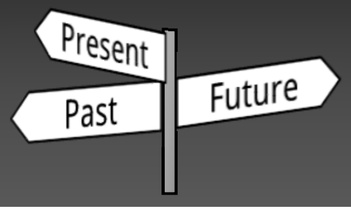
Taint Analysis



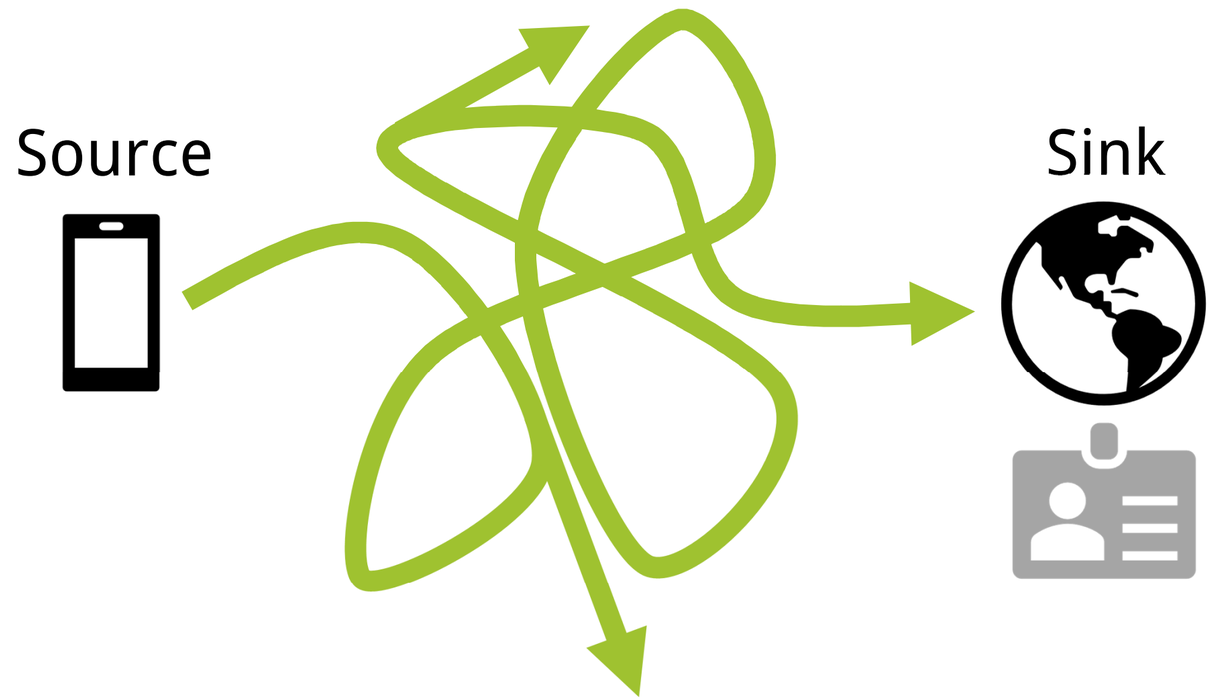
Tools



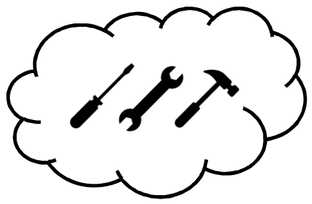
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- …



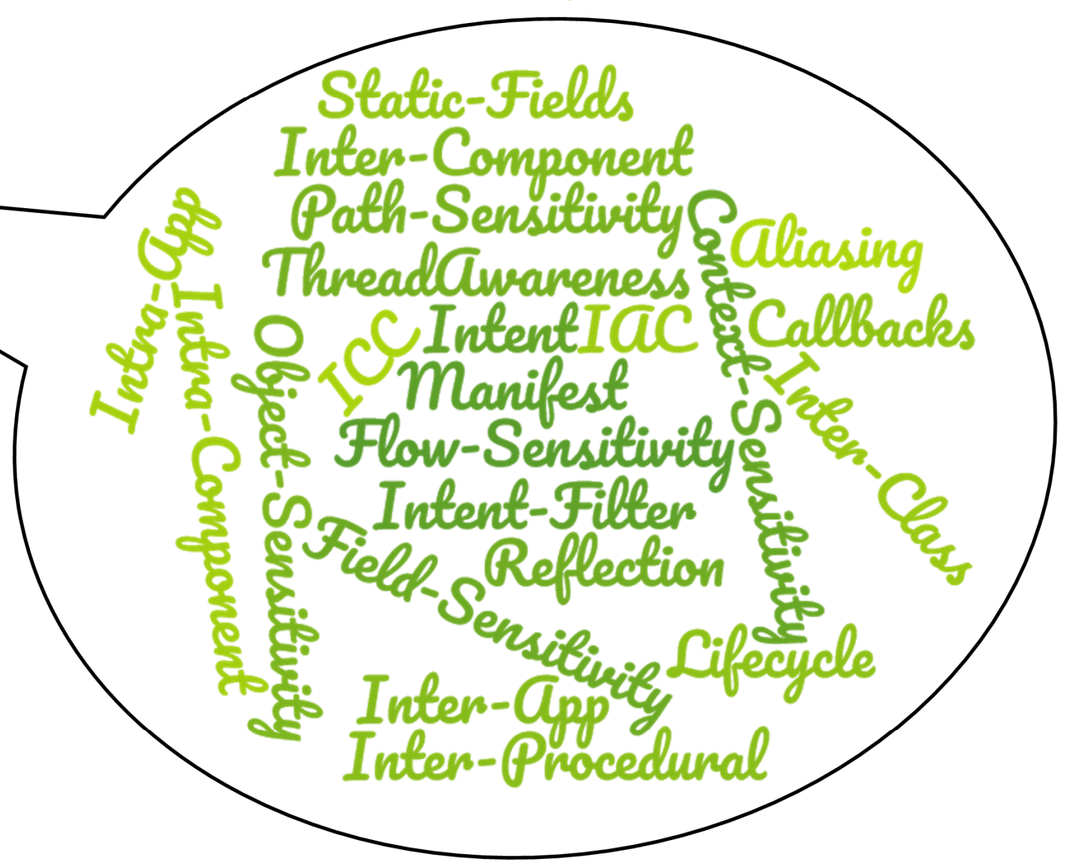
Taint Analysis

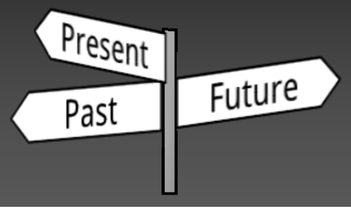


Tools

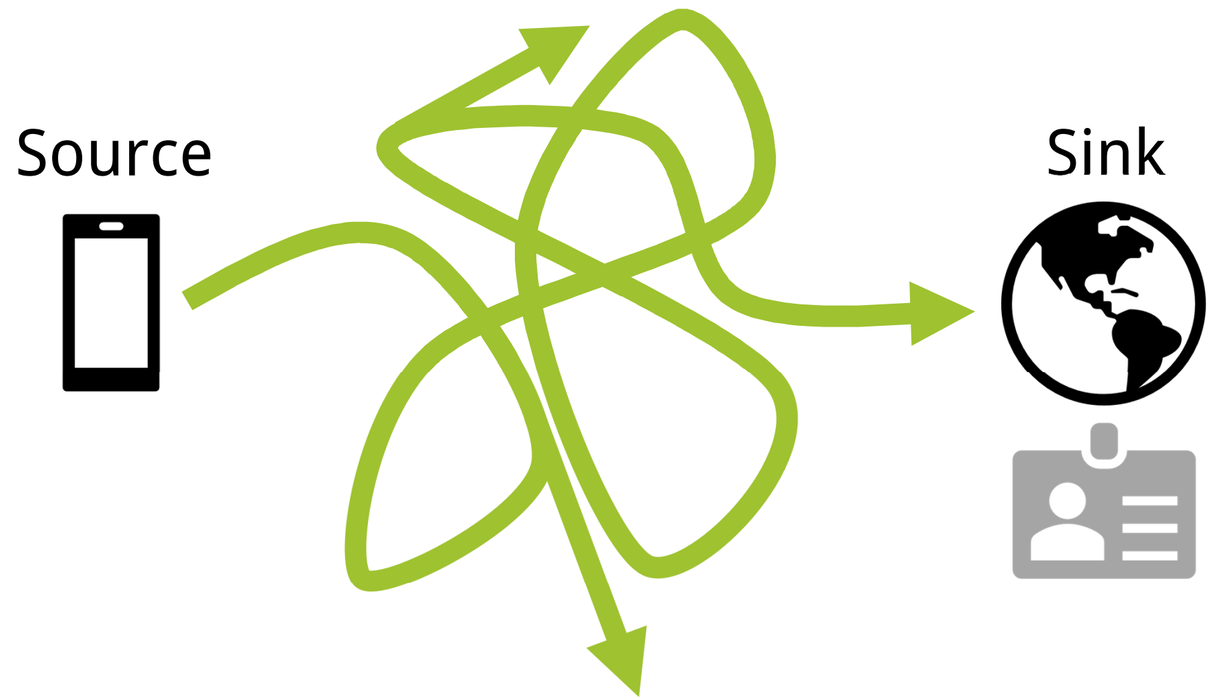


- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- …

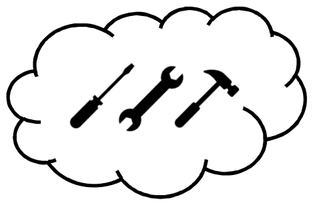




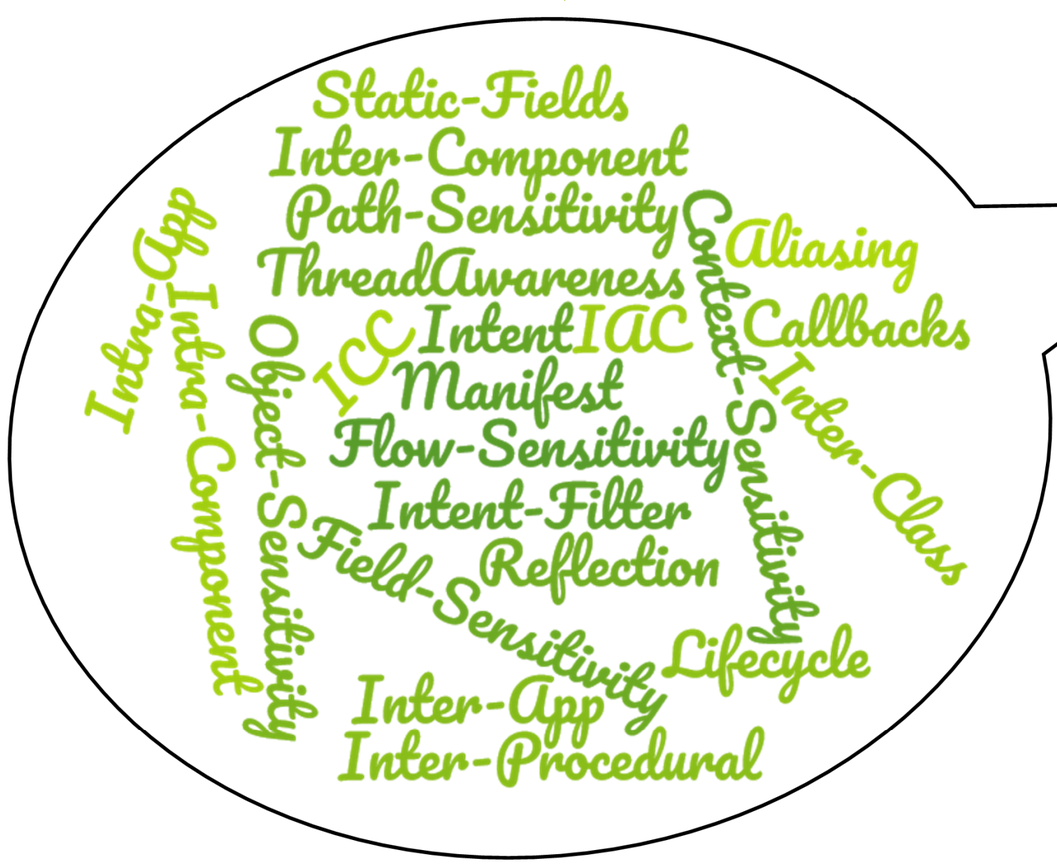
Taint Analysis



Tools

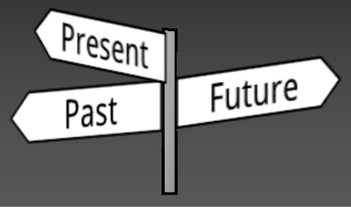


- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- …

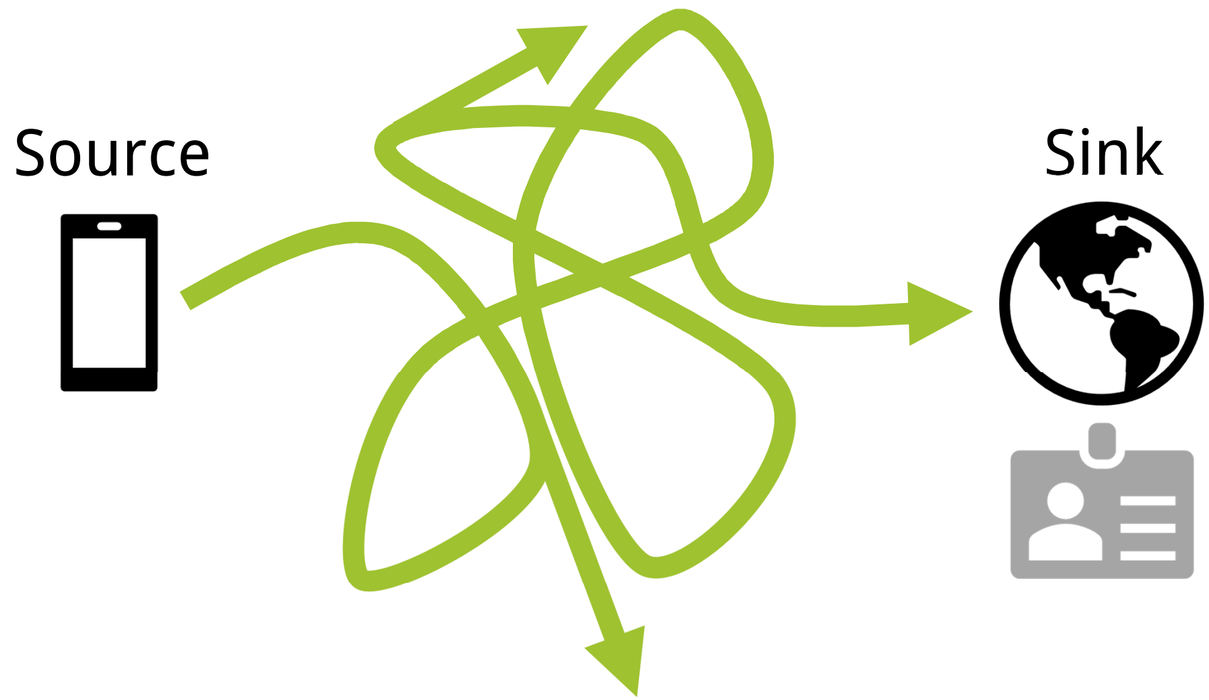


Benchmarks

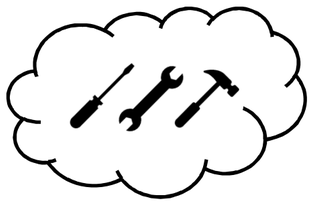
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …



Taint Analysis



Tools

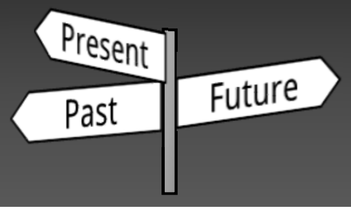


- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA
- …

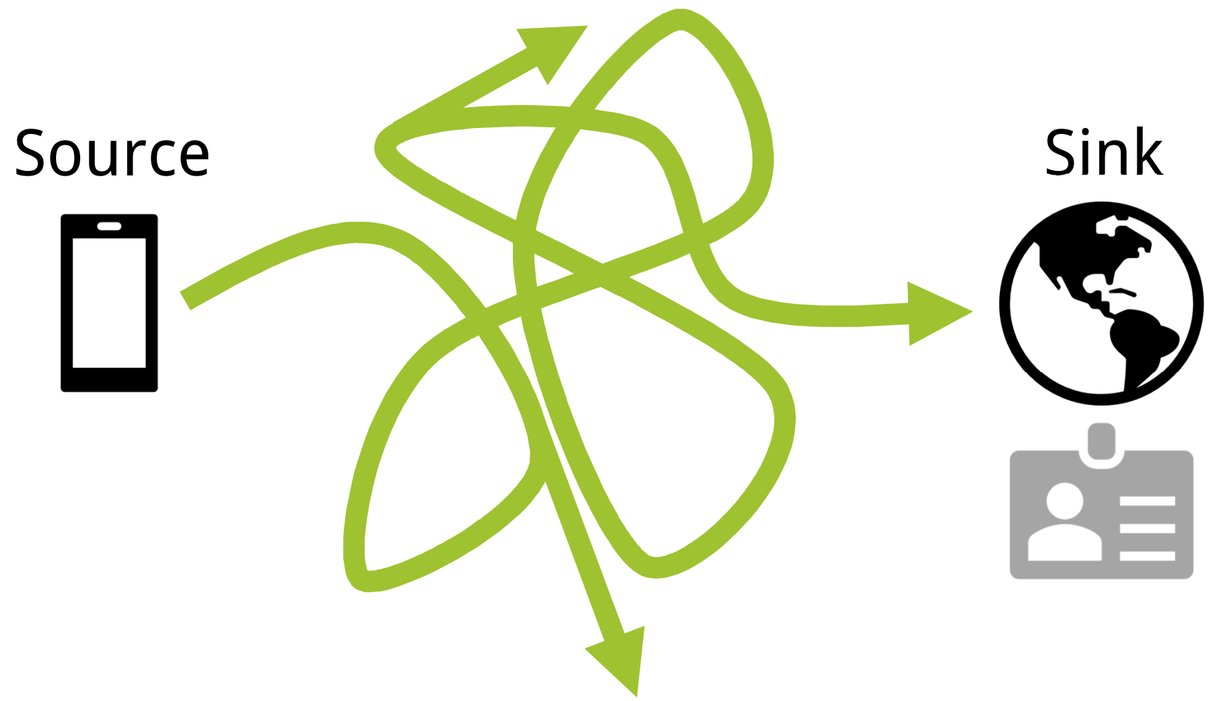
Actual results

Benchmarks

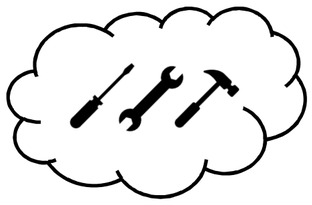
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …



Taint Analysis



Tools



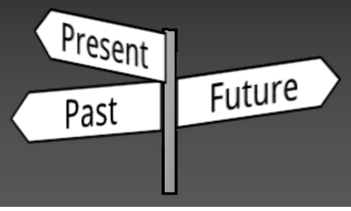
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA

Actual results

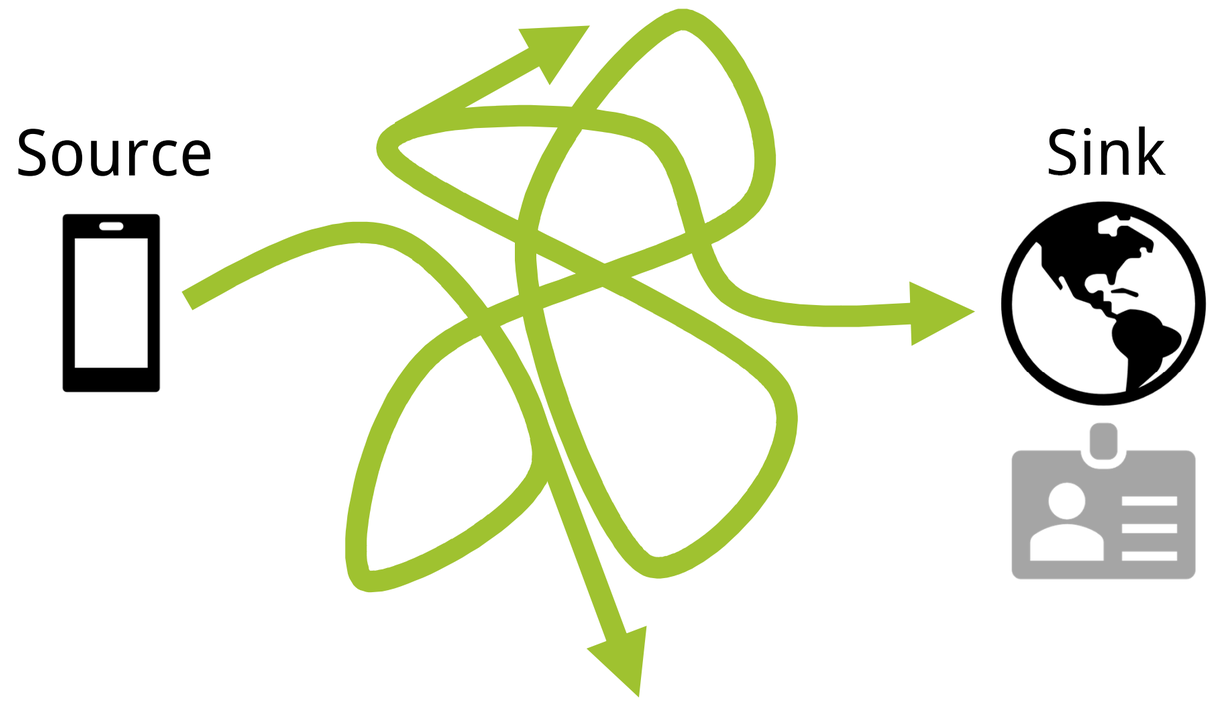
Benchmarks

Ground-truth / Expected results

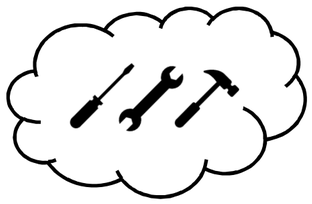
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …



Taint Analysis



Tools

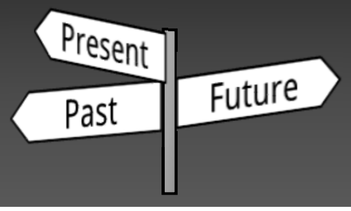


- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA

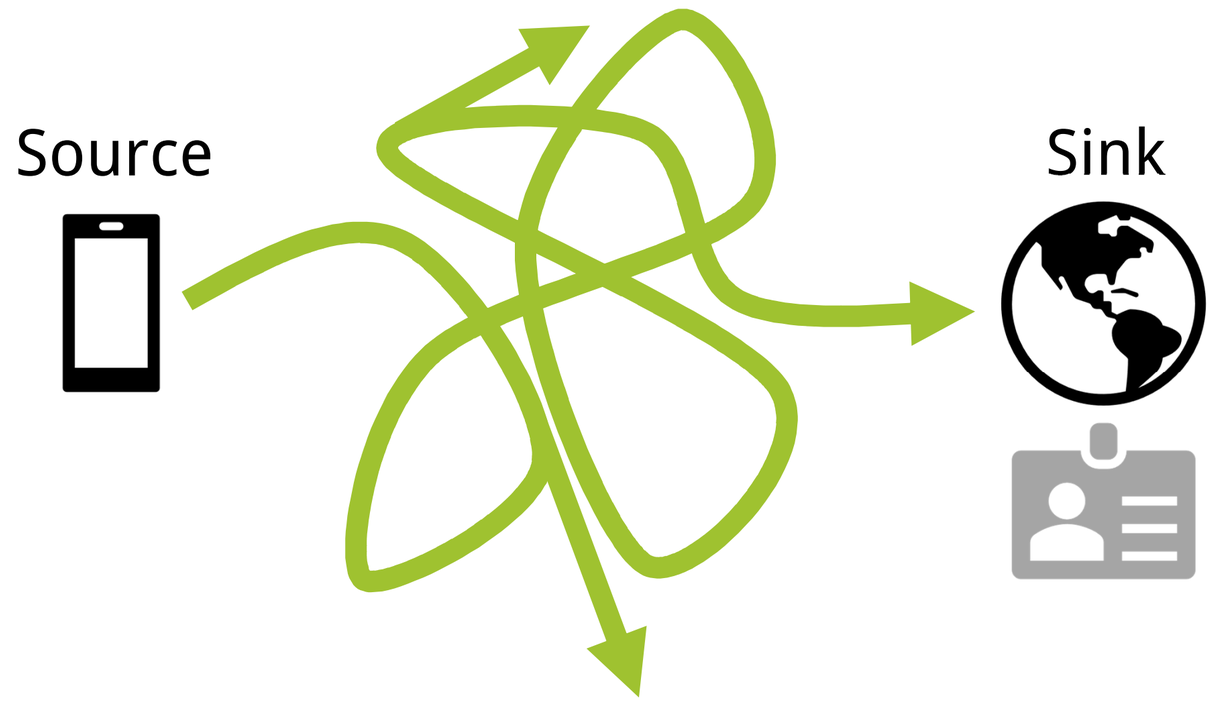
Actual results =? Ground-truth / Expected results

Benchmarks

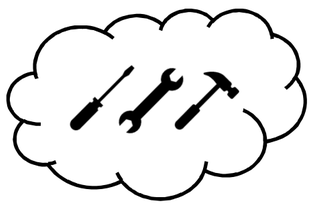
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …



Taint Analysis



Tools



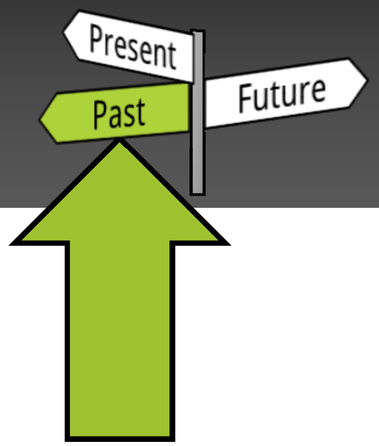
- Amandroid
- DIALDroid
- DidFail
- DroidSafe
- FlowDroid
- IccTA

Actual results

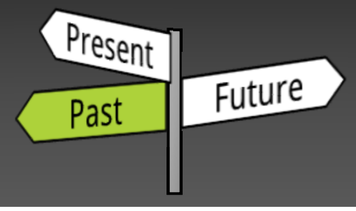
Ground-truth / Expected results

Benchmarks

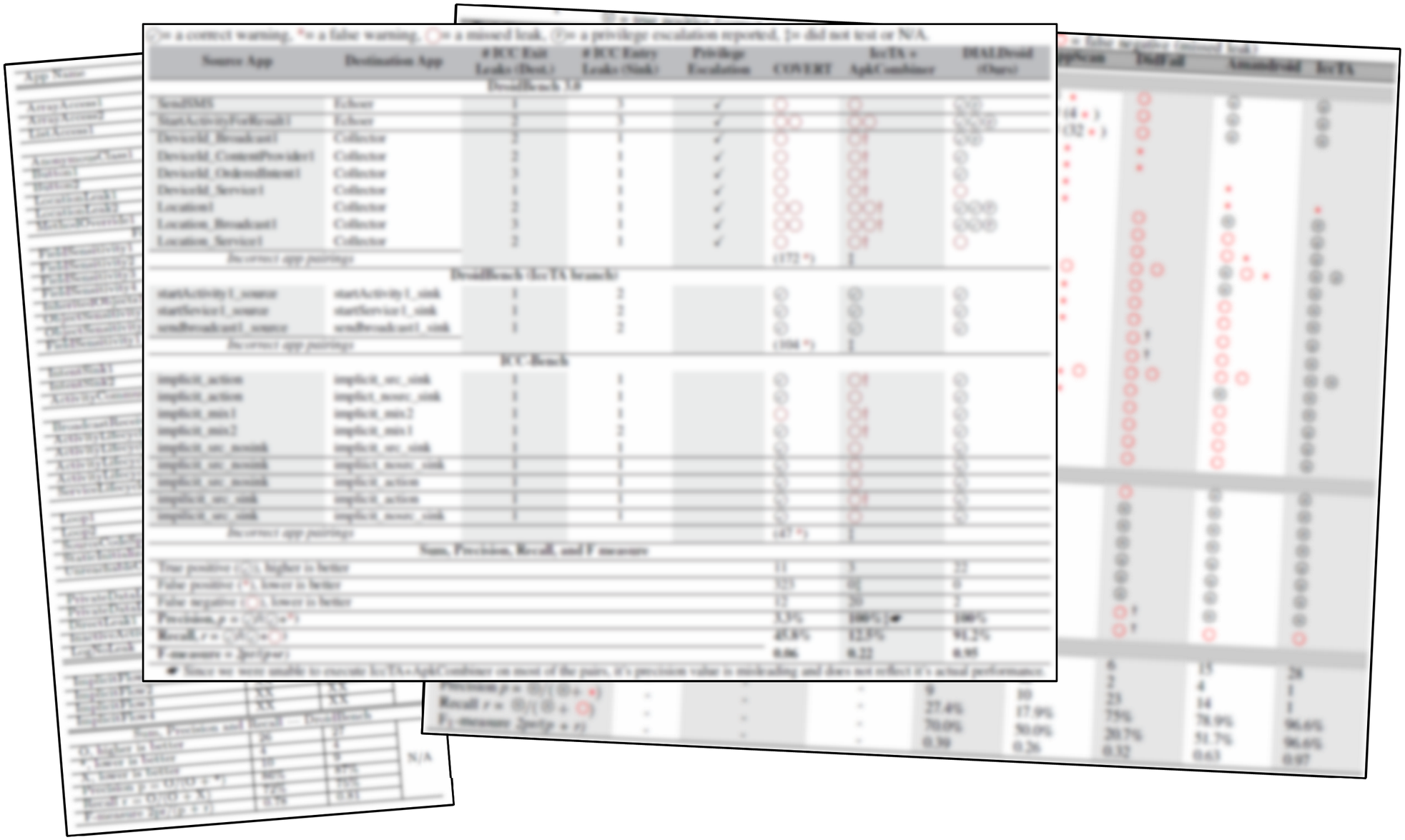
- DroidBench 3.0
- DroidBench 2.0
- ICCBench 2.0
- …



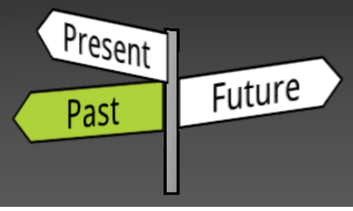
Benchmarks: Past



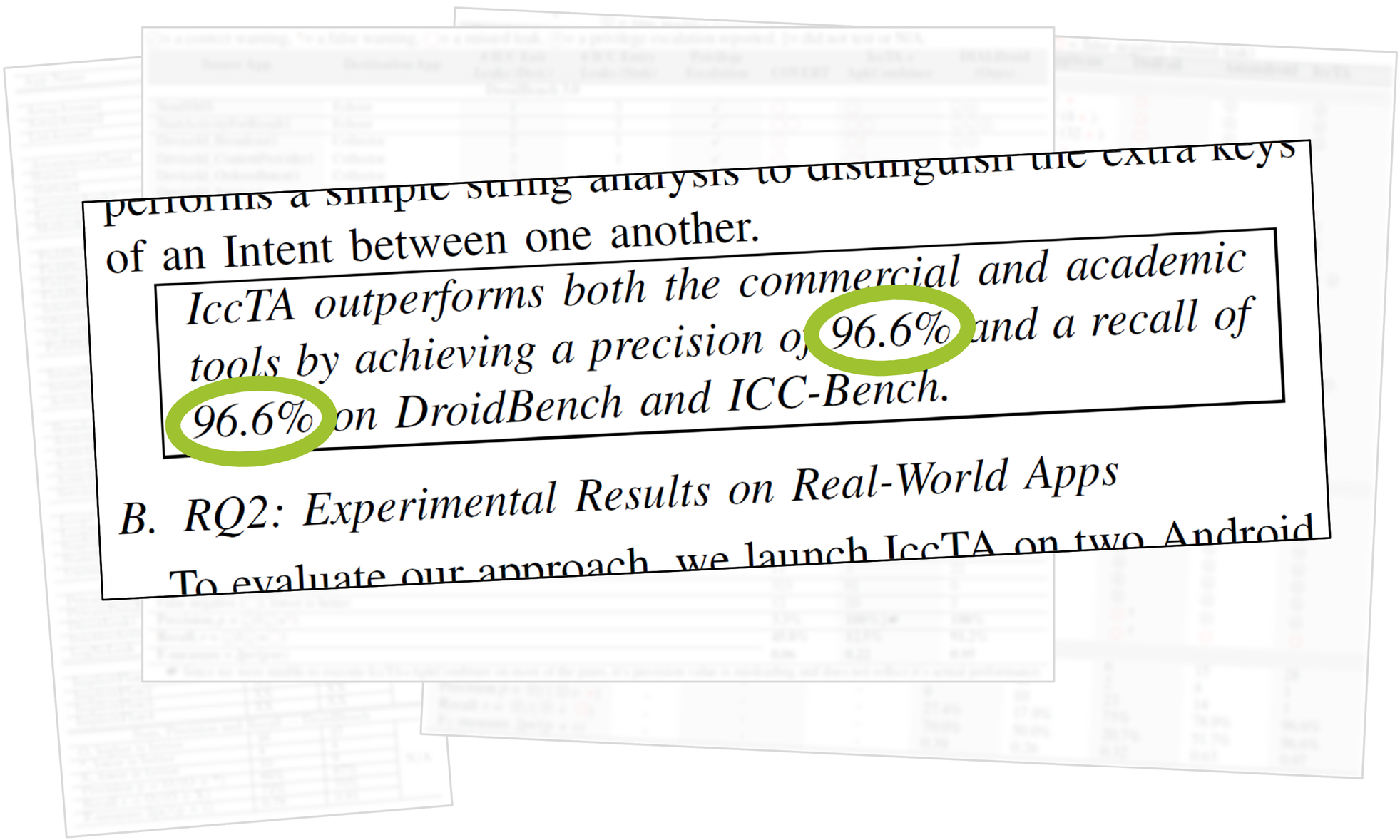
Benchmarks: Past



1) Fengguo Wei, Sankardas Roy, Xinming Ou, and Robby. 2014. Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (CCS '14). ACM, New York, NY, USA, 1329-1341. DOI: http://dx.doi.org/10.1145/2660267.2660357
2) Li Li, Alexandre Bartel, Tegawendé F. Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. 2015. IccTA: detecting inter-component privacy leaks in Android apps. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (ICSE '15), Vol. 1. IEEE Press, Piscataway, NJ, USA, 280-291.
3) Amiangshu Bosu, Fang Liu, Danfeng (Daphne) Yao, and Gang Wang. 2017. Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (ASIA CCS '17). ACM, New York, NY, USA, 71-85. DOI: https://doi.org/10.1145/3052973.3053004



Benchmarks: Past



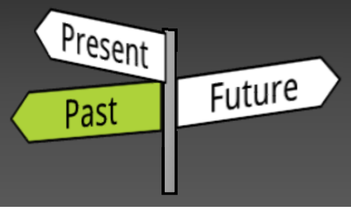
of an Intent between one another.

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.

B. RQ2: Experimental Results on Real-World Apps

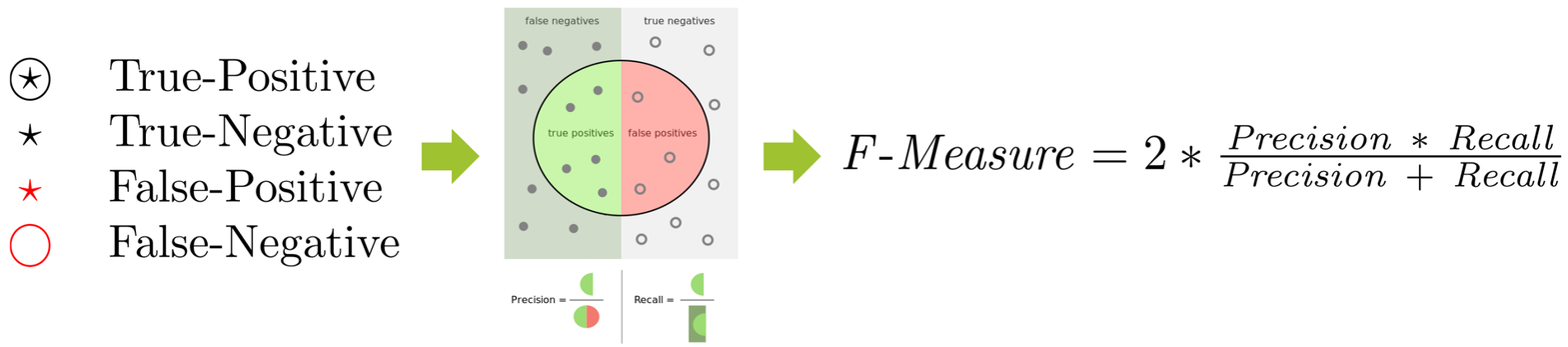
To evaluate our approach, we launch IccTA on two Android

1) Fengguo Wei, Sankardas Roy, Xinming Ou, and Robby. 2014. Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (CCS '14). ACM, New York, NY, USA, 1329-1341. DOI: http://dx.doi.org/10.1145/2660267.2660357
2) Li Li, Alexandre Bartel, Tegawendé F. Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. 2015. IccTA: detecting inter-component privacy leaks in Android apps. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (ICSE '15), Vol. 1. IEEE Press, Piscataway, NJ, USA, 280-291.
3) Amiangshu Bosu, Fang Liu, Danfeng (Daphne) Yao, and Gang Wang. 2017. Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (ASIA CCS '17). ACM, New York, NY, USA, 71-85. DOI: https://doi.org/10.1145/3052973.3053004



Benchmarks: Past

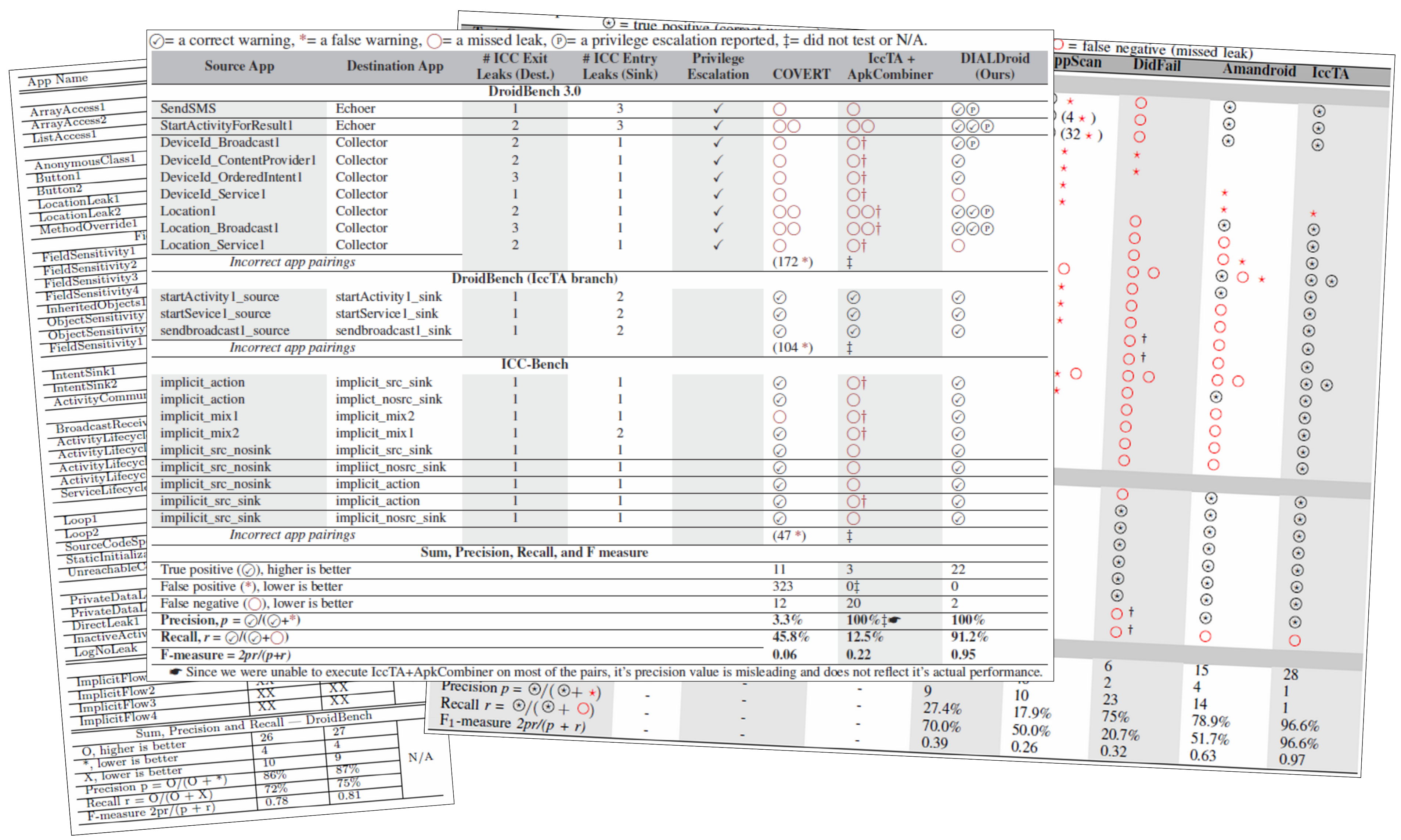
- Standardized evaluation procedure,



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: Past

- Standardized evaluation procedure,
- Large userbase,

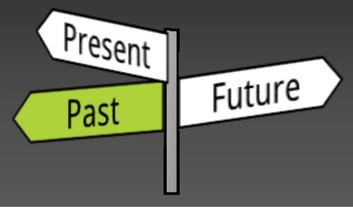


(✓)= a correct warning, *= a false warning, ○= a missed leak, Ⓟ= a privilege escalation reported, ‡= did not test or N/A.

Source App	Destination App	# ICC Exit Leaks (Dest.)	# ICC Entry Leaks (Sink)	Privilege Escalation	COVERT	IccTA + ApkCombiner	DIALDroid (Ours)
DroidBench 3.0							
SendSMS	Echoer	1	3	✓	○	○	(✓)Ⓟ
StartActivityForResult1	Echoer	2	3	✓	○○	○○	(✓)(✓)Ⓟ
DeviceId_Broadcast1	Collector	2	1	✓	○	○†	(✓)Ⓟ
DeviceId_ContentProvider1	Collector	2	1	✓	○	○†	(✓)
DeviceId_OrderedIntent1	Collector	3	1	✓	○	○†	(✓)
DeviceId_Service1	Collector	1	1	✓	○	○†	○
Location1	Collector	2	1	✓	○○	○○†	(✓)(✓)Ⓟ
Location_Broadcast1	Collector	3	1	✓	○○	○○†	(✓)(✓)Ⓟ
Location_Service1	Collector	2	1	✓	○	○†	○
Incorrect app pairings					(172 *)	‡	
DroidBench (IccTA branch)							
startActivity1_source	startActivity1_sink	1	2		(✓)	(✓)	(✓)
startSevice1_source	startService1_sink	1	2		(✓)	(✓)	(✓)
sendbroadcast1_source	sendbroadcast1_sink	1	2		(✓)	(✓)	(✓)
Incorrect app pairings					(104 *)	‡	
ICC-Bench							
implicit_action	implicit_src_sink	1	1		(✓)	○†	(✓)
implicit_action	implict_nosrc_sink	1	1		(✓)	○	(✓)
implicit_mix1	implicit_mix2	1	1		○	○†	(✓)
implicit_mix2	implicit_mix1	1	2		(✓)	○†	(✓)
implicit_src_nosink	implicit_src_sink	1	1		(✓)	○	(✓)
implicit_src_nosink	impliict_nosrc_sink	1	1		(✓)	○	(✓)
implicit_src_nosink	implicit_action	1	1		(✓)	○	(✓)
impilicit_src_sink	implicit_action	1	1		(✓)	○†	(✓)
impilicit_src_sink	implicit_nosrc_sink	1	1		(✓)	○	(✓)
Incorrect app pairings					(47 *)	‡	
Sum, Precision, Recall, and F measure							
True positive ((✓)), higher is better					11	3	22
False positive (*), lower is better					323	0‡	0
False negative (○), lower is better					12	20	2
Precision, p = (✓)/((✓)+*)					**3.3%**	**100%‡☛**	**100%**
Recall, r = (✓)/((✓)+○)					**45.8%**	**12.5%**	**91.2%**
F-measure = ***2pr/(p+r)***					**0.06**	**0.22**	**0.95**

☛ Since we were unable to execute IccTA+ApkCombiner on most of the pairs, it's precision value is misleading and does not reflect it's actual performance.

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: Past

- \+ Standardized evaluation procedure,
- \+ Large userbase,
- − A collection of deprecated apps,

eclipse-project →

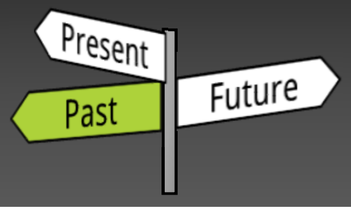
3 years ago

apk	added a new test case	3 years ago
eclipse-project	added a new test case	3 years ago
.gitignore	fixed permissions in the existing exception test cases and added two ...	3 years ago
JavaDoc_Template.txt	javadoc_template first version	6 years ago
README.md	fixed the number of test cases	3 years ago
droidbench_apps.png	added yet another test case and fixed to logo URL	3 years ago
new.gif	added new gif	6 years ago

README.md

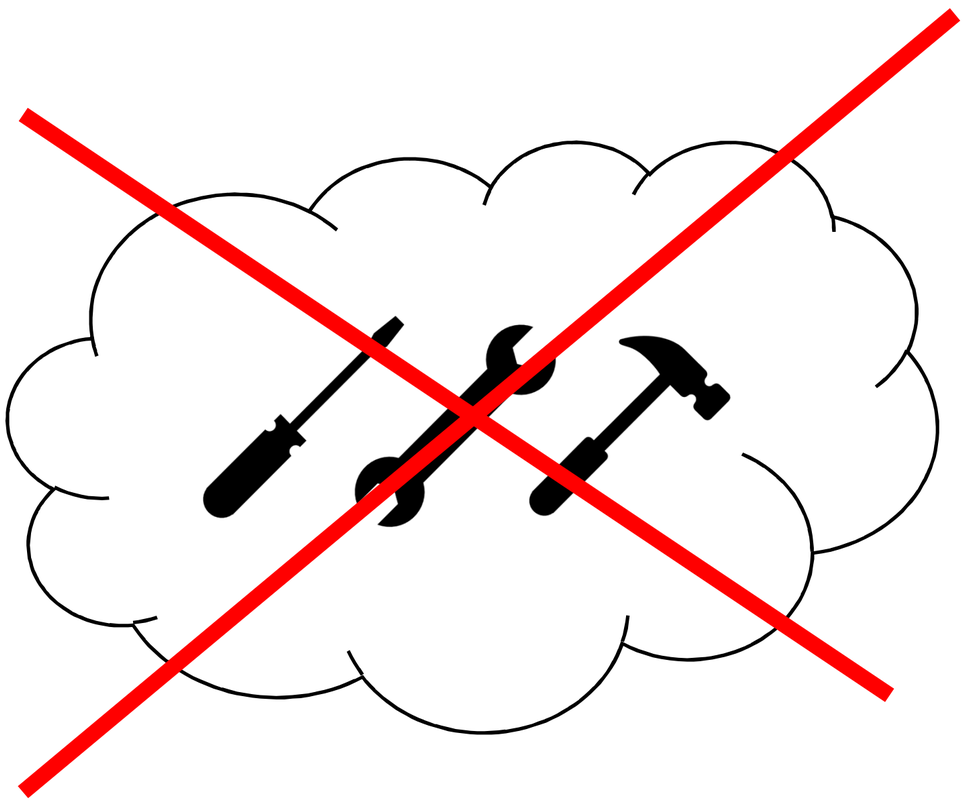
DroidBench 3.0

https://en.wikipedia.org/wiki/Precision_and_recall
https://github.com/secure-software-engineering/DroidBench/tree/develop
https://en.wikipedia.org/wiki/Benchmark_(computing)

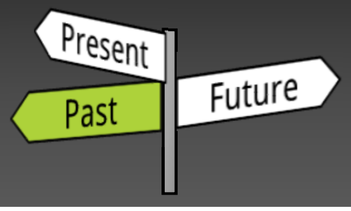


Benchmarks: Past

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



https://en.wikipedia.org/wiki/Precision_and_recall
https://github.com/secure-software-engineering/DroidBench/tree/develop
https://en.wikipedia.org/wiki/Benchmark_(computing)

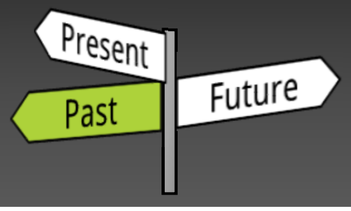


Benchmarks: Past

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

```
12  import android.widget.Button;
13
14  /**
15   * @testcase_name StartActivityForResult1
16   * @version 0.1
17   * @author Contributed by
18   * @author_mail (Maintainer)
19   *
20   * @description Reads the user's geographical location (via GPS) and leaks
21   *              it to the file system, and passes it to another activity using
22   *              startActivityForResult which writes it to a file.
23   * @dataflow getLastKnownLocation -> startActivityForResult
24   *              -> onActivityResult -> FileOutputStream
25   * @number_of_leak 1
26   * @challenges Inter-component communication using startActivityForResult
27   *              must be handled correctly
28   */
29  public class MainActivity extends Activity {
```

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: Past

\+ Standardized evaluation procedure,

\+ Large userbase,

− A collection of deprecated apps,

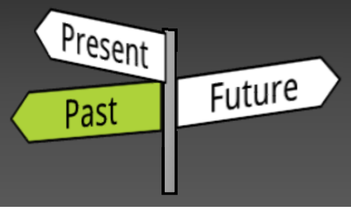
− Not executable,

− Manual evaluation,

− No reproducibility.

"The term benchmark is utilized for the benchmarking programs themselves"*

* 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: Past

\+ Standardized evaluation procedure,

\+ Large userbase,

− A collection of deprecated apps,

− Not executable,

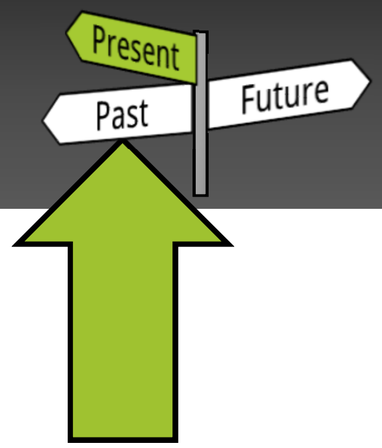
− Manual evaluation,

− No reproducibility.

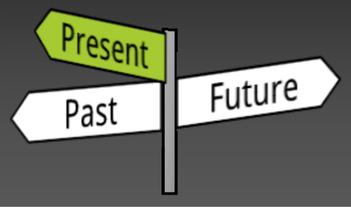
"The term benchmark is utilized for the benchmarking programs themselves"*

"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)

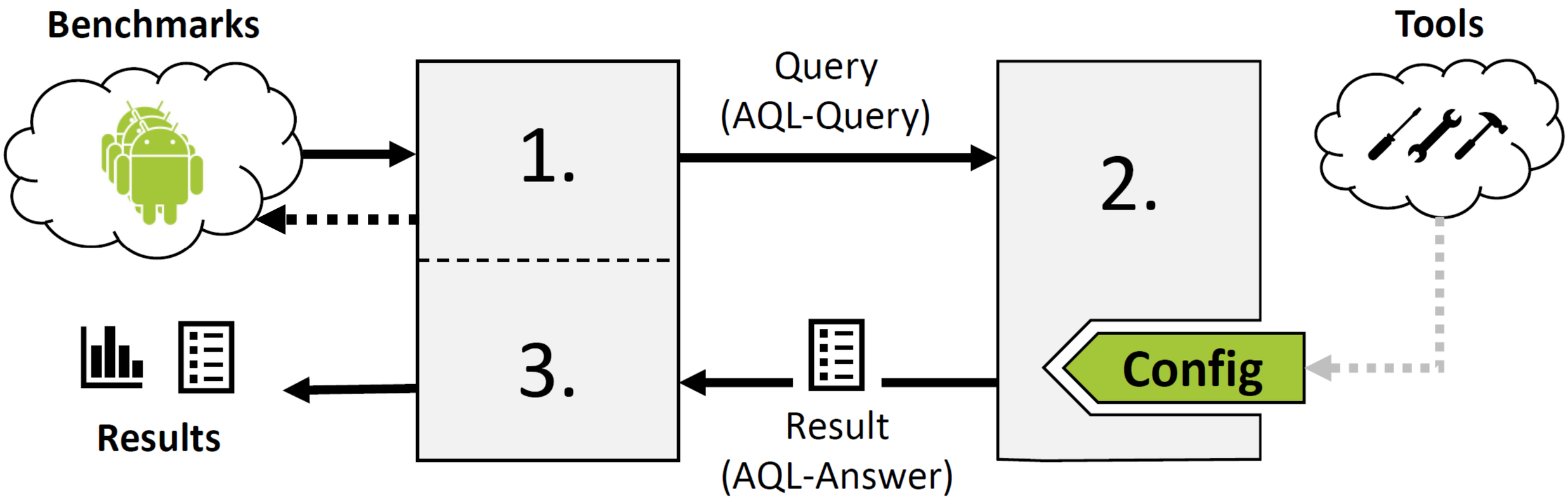


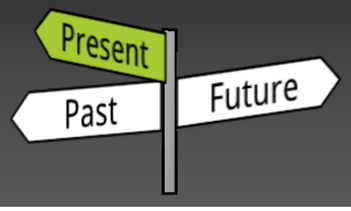
Benchmarks: Present



Benchmarks: Present

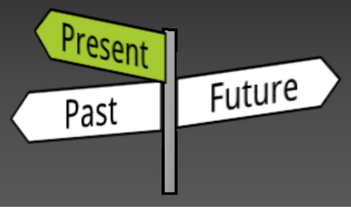
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]





Benchmarks: Present

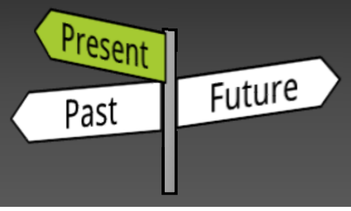
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]



Benchmarks: Present

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

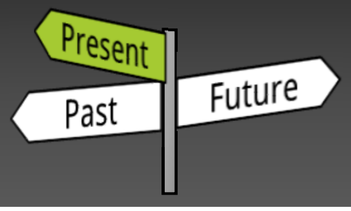
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)



Benchmarks: Present

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

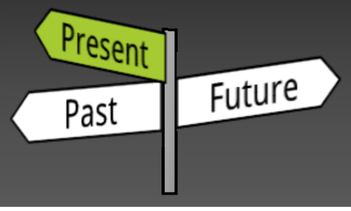
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)



Benchmarks: Present

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

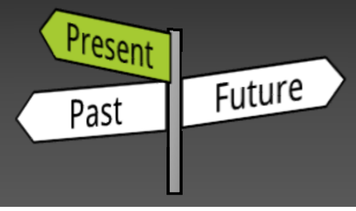
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)



Benchmarks: Present

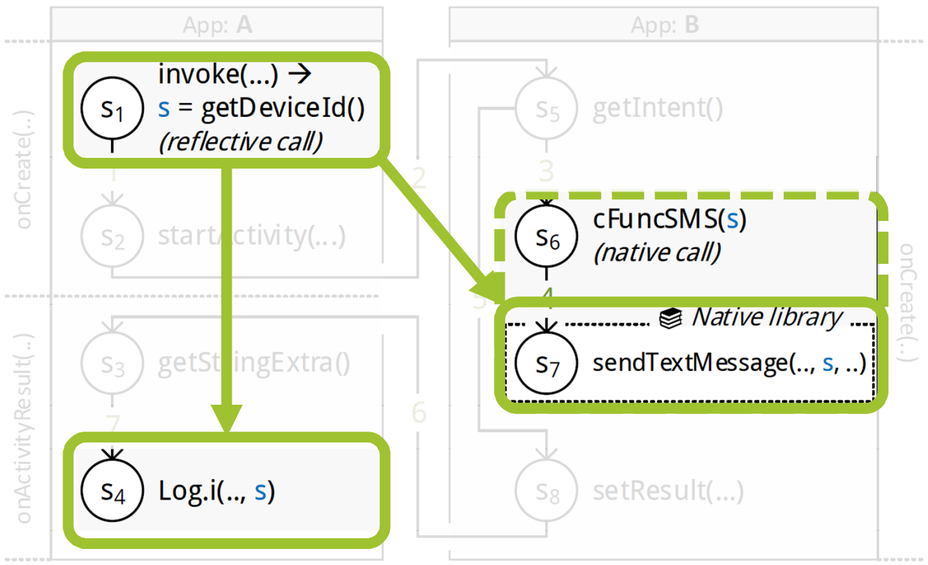
Do Android Taint Analysis tools keep their promises? [ESEC/FSE'18]

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



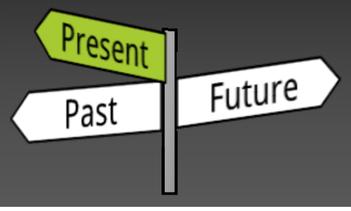
Benchmarks: Present

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



```
<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>
```

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



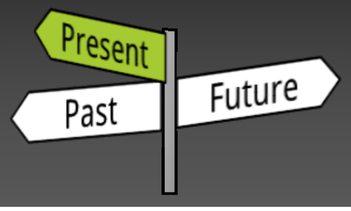
Benchmarks: Present

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

Config

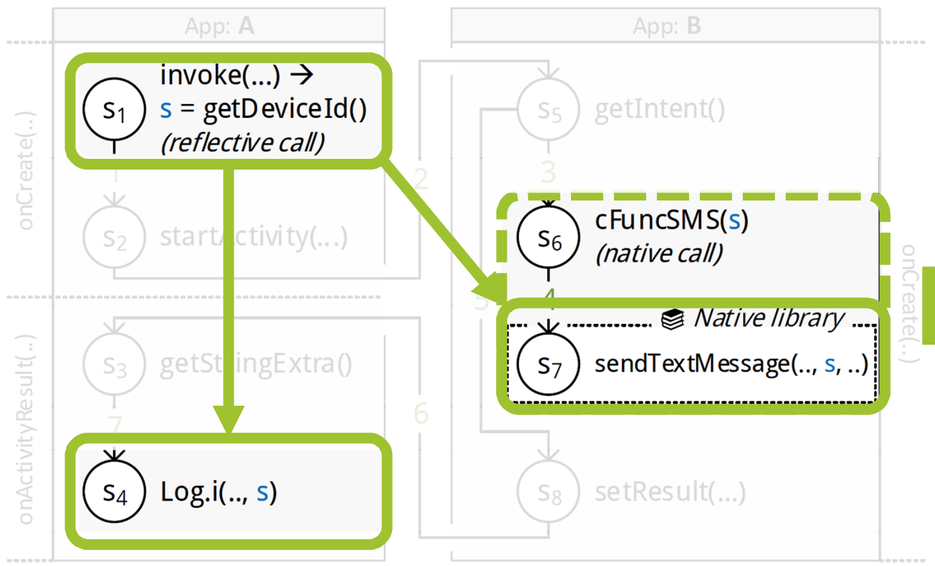
```
Flows FROM App('A.apk') TO App('B.apk') ?

-----------------------------------------------------------

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') ?
]
```

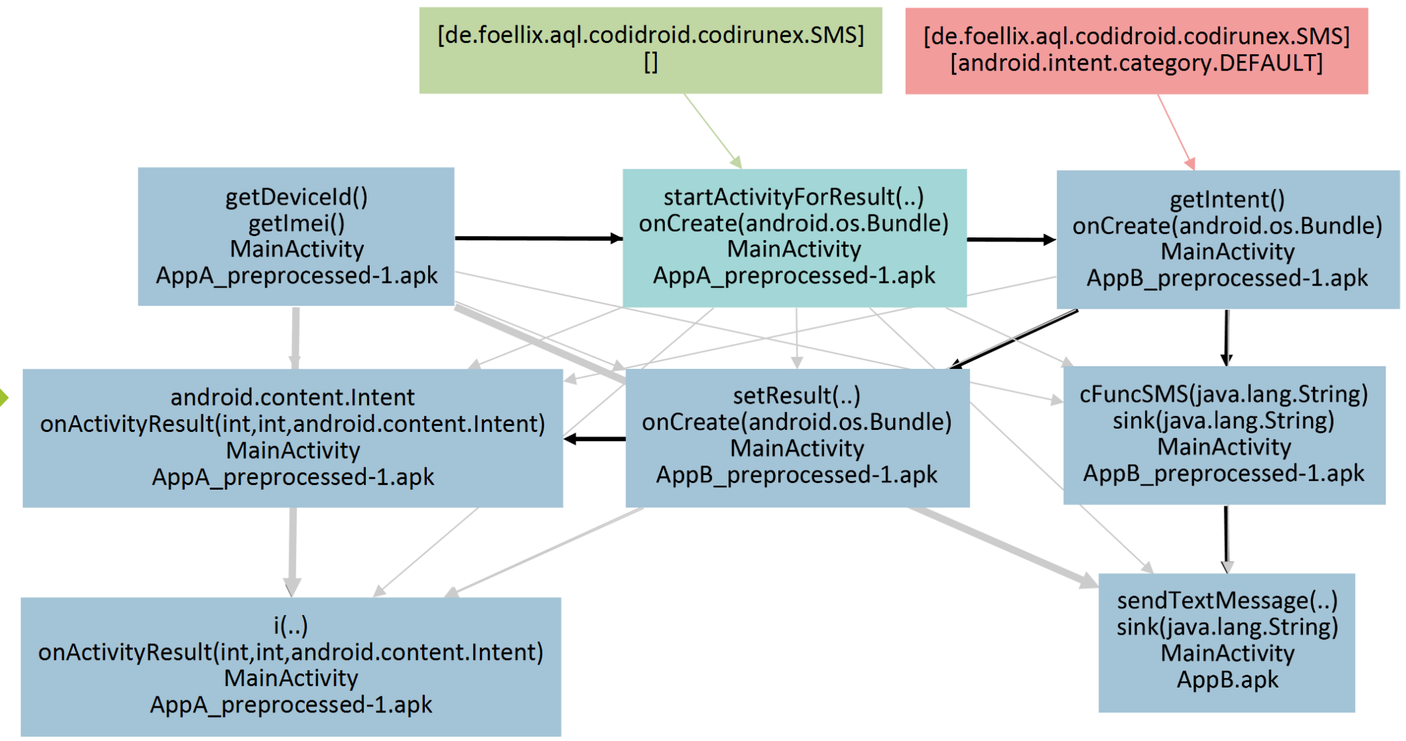
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)



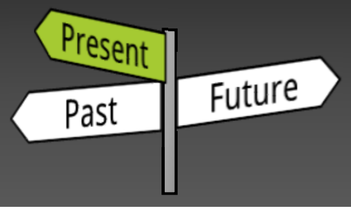
Benchmarks: Present

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



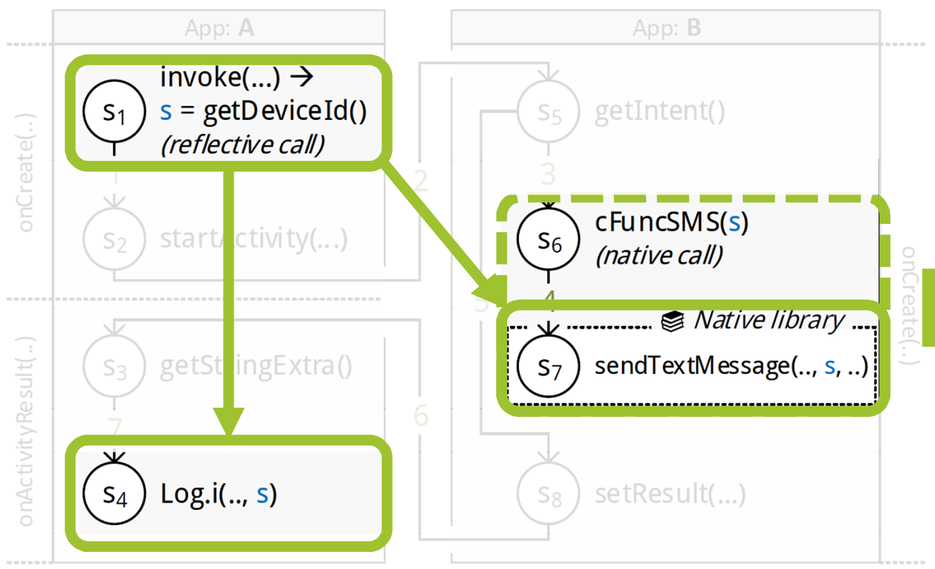


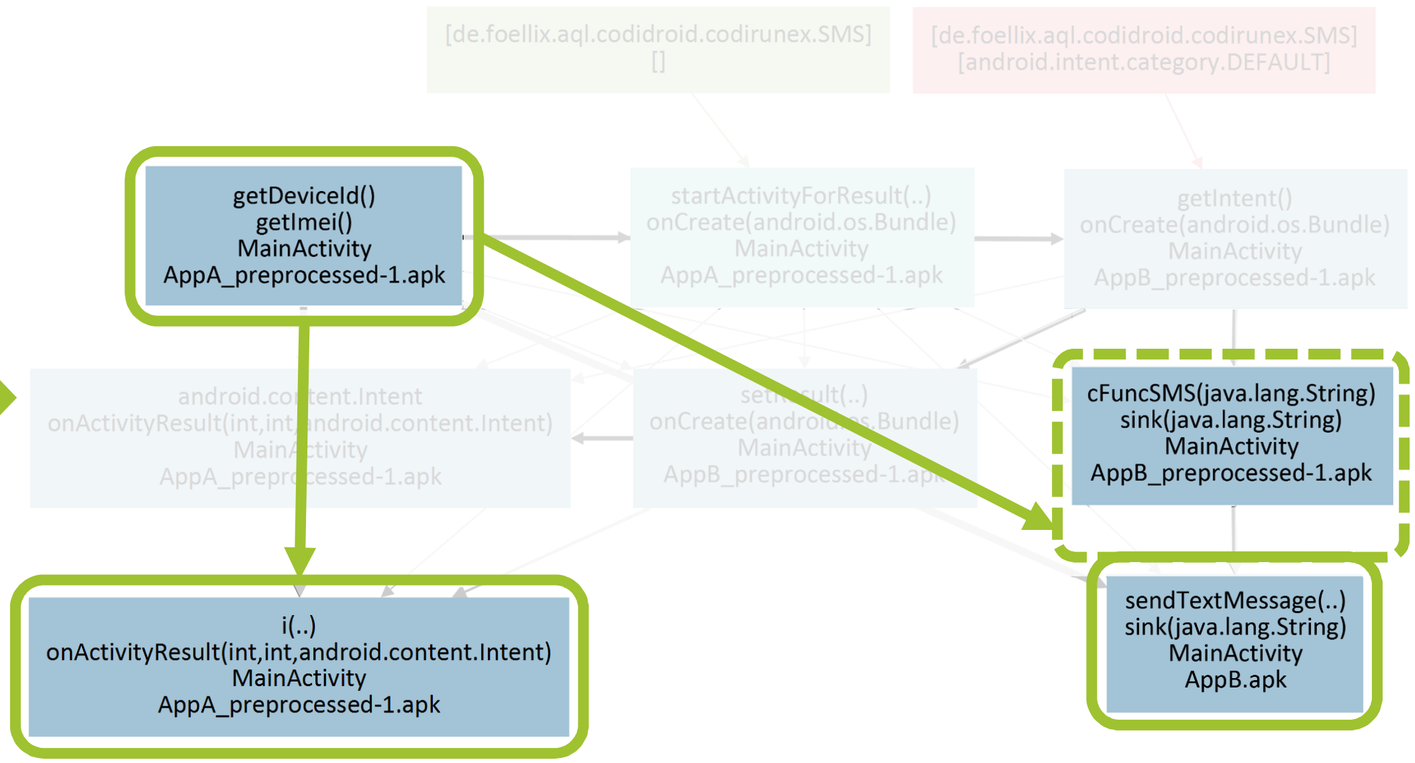
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)



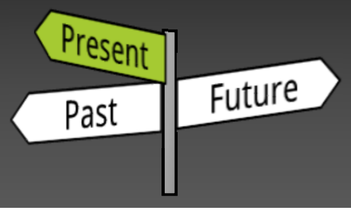
Benchmarks: Present

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



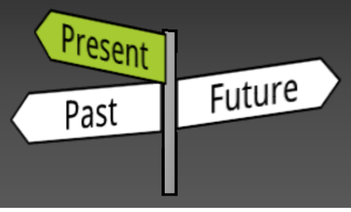


1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)



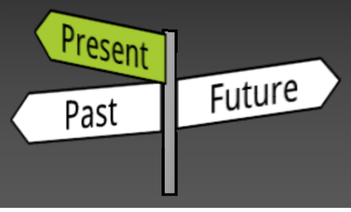
Benchmarks: Present

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



Benchmarks: Present

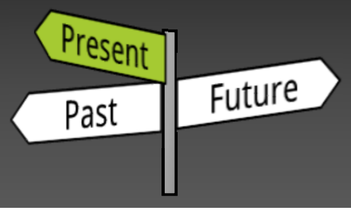
- \+ Standardized evaluation procedure,
- \+ Large userbase,
- − A collection of deprecated apps,
- \+ Executable,
- \+ Automatic evaluation,
- \+ Reproducible.



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,

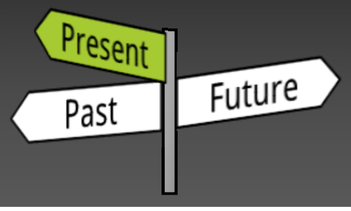
\+ Executable,

\+ Automatic evaluation,

\+ Reproducible.

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

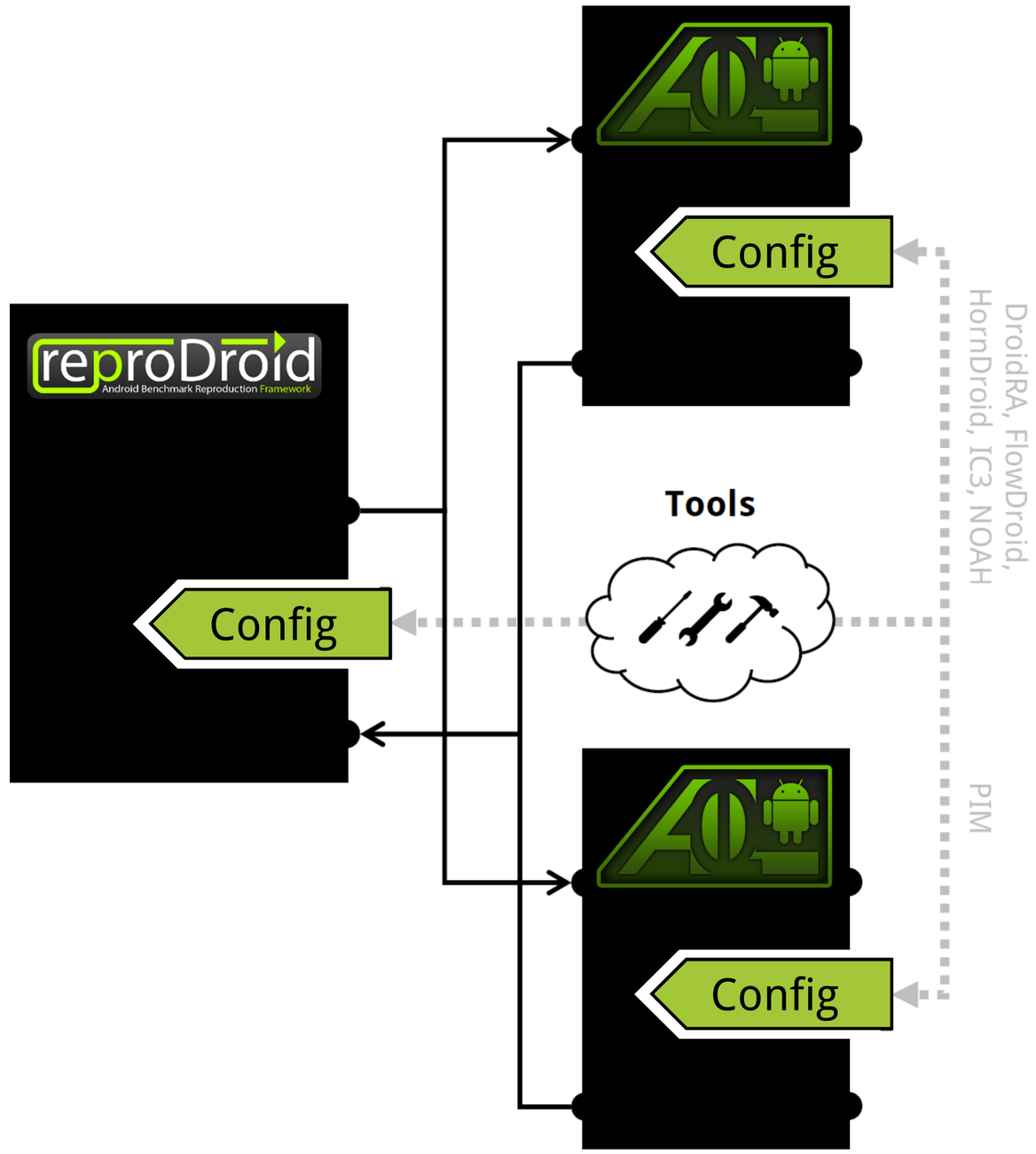
“a benchmark is the act of running a program in order to assess the relative performance”*

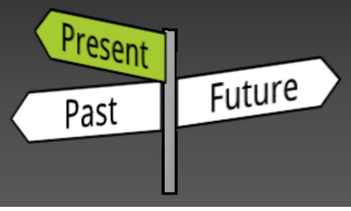


Benchmarks: Present

- Evaluate novel tools

CoDiDroid
together strong!

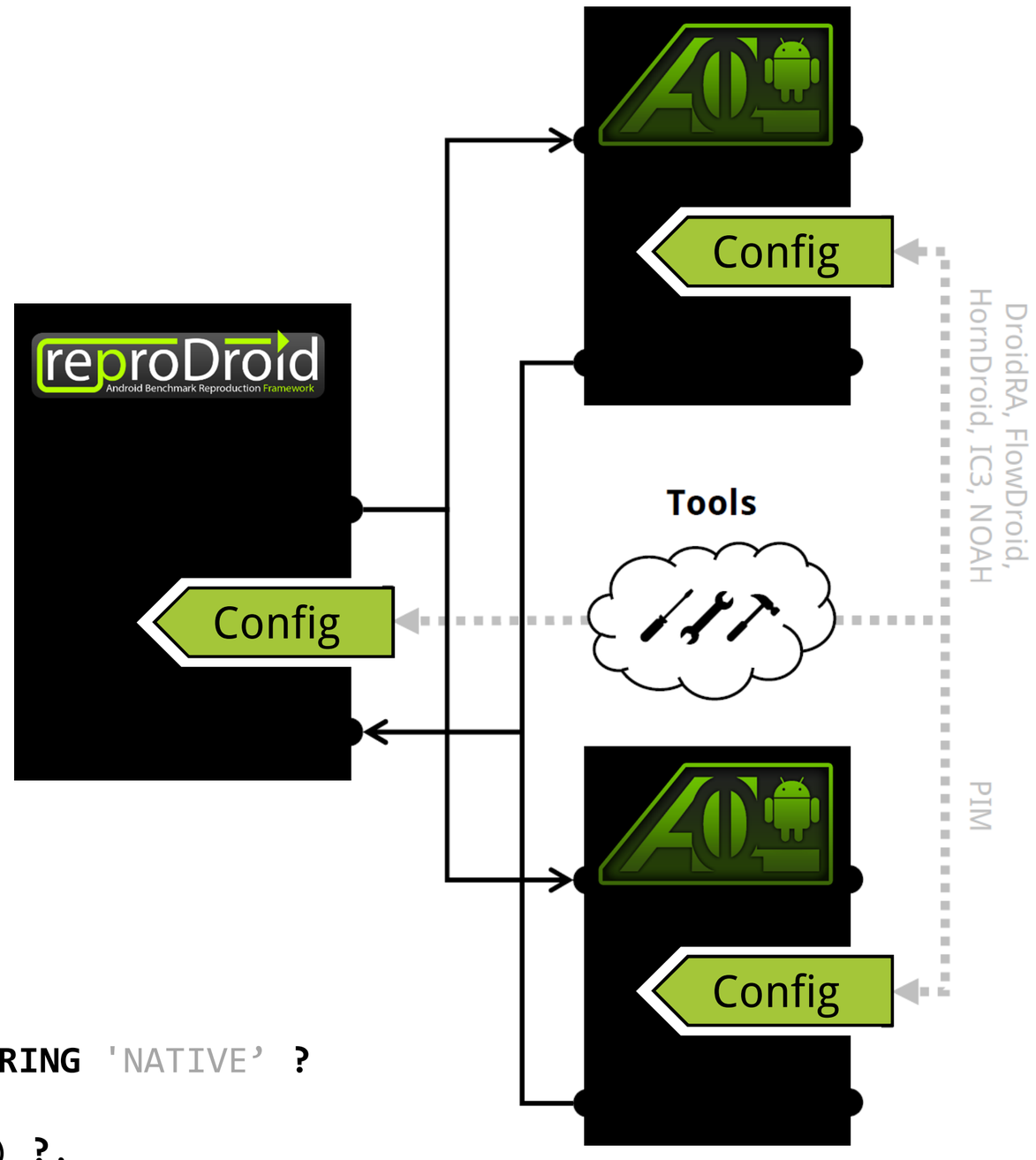




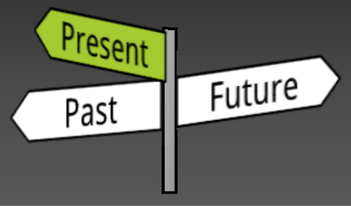
Benchmarks: Present

- Evaluate novel tools

CoDiDroid
together strong!




```
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') ?
]
```

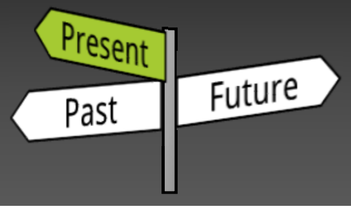


Benchmarks: Present

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

- Evaluate nov

ID	Category	FlowDroid	Best	CoDiDroid	Difference to Best	Difference to FlowDroid
1	Aliasing					
2	AndroidSpecific					
3	ArraysAndLists					
4	Callbacks					
5	DynamicLoading					
6	EmulatorDetection					
7	FieldAndObjectSensitivity					
8	GeneralJava					
9	ImplicitFlows					
10	InterAppCommunication					
11	InterComponentCommunication					
12	Lifecycle					
13	Native					
14	Reflection					
15	Reflection_ICC					
16	SelfModification					
17	Threading					
18	UnreachableCode					
	Ø					

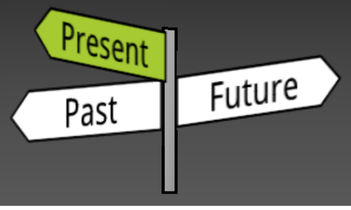


Benchmarks: Present

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

- Evaluate nov

ID	Category	FlowDroid	Best	CoDiDroid	Difference to Best	Difference to FlowDroid
1	Aliasing	0.667				
2	AndroidSpecific	0.900				
3	ArraysAndLists	0.615				
4	Callbacks	0.897				
5	DynamicLoading	0.000				
6	EmulatorDetection	0.966				
7	FieldAndObjectSensitivity	1.000				
8	GeneralJava	0.810				
9	ImplicitFlows	0.000				
10	InterAppCommunication	0.000				
11	InterComponentCommunication	0.348				
12	Lifecycle	0.769				
13	Native	0.000				
14	Reflection	0.095				
15	Reflection_ICC	0.000				
16	SelfModification	0.000				
17	Threading	1.000				
18	UnreachableCode	1.000				
	Ø	0.504				

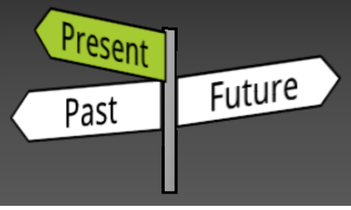


Benchmarks: Present

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

- Evaluate nov

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

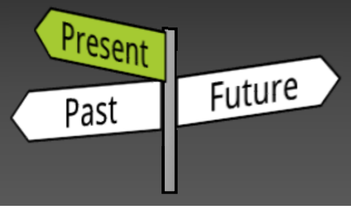


Benchmarks: Present

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

- Evaluate nov

ID	Category	FlowDroid	Best	CoDiDroid	Difference to Best	Difference to FlowDroid
1	Aliasing	0.667		0.667		0.000
2	AndroidSpecific	0.900		0.900		0.000
3	ArraysAndLists	0.615		0.615		0.000
4	Callbacks	0.897		0.897		0.000
5	DynamicLoading	0.000		0.000		0.000
6	EmulatorDetection	0.966		0.966		0.000
7	FieldAndObjectSensitivity	1.000		1.000		0.000
8	GeneralJava	0.810		0.810		0.000
9	ImplicitFlows	0.000		0.000		0.000
10	InterAppCommunication	0.000		0.625		-0.625
11	InterComponentCommunication	0.348		0.690		-0.342
12	Lifecycle	0.769		0.769		0.000
13	Native	0.000		0.889		-0.889
14	Reflection	0.095		0.800		-0.705
15	Reflection_ICC	0.000		0.000		0.000
16	SelfModification	0.000		0.000		0.000
17	Threading	1.000		1.000		0.000
18	UnreachableCode	1.000		1.000		0.000
	Ø	0.504		0.646		-0.142

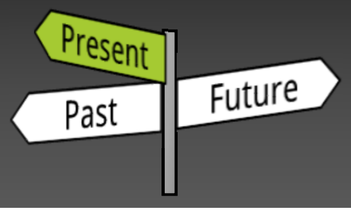


Benchmarks: Present

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

- Evaluate nov

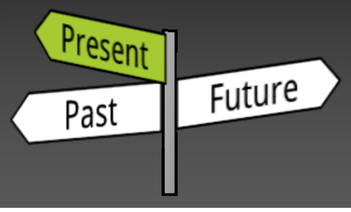
ID	Category	FlowDroid	Best	CoDiDroid	Difference to Best	Difference to FlowDroid
1	Aliasing	0.667	0.667	0.667	0.000	0.000
2	AndroidSpecific	0.900	0.900	0.900	0.000	0.000
3	ArraysAndLists	0.615	0.667	0.615	0.052	0.000
4	Callbacks	0.897	0.897	0.897	0.000	0.000
5	DynamicLoading	0.000	0.500	0.000	0.500	0.000
6	EmulatorDetection	0.966	0.966	0.966	0.000	0.000
7	FieldAndObjectSensitivity	1.000	1.000	1.000	0.000	0.000
8	GeneralJava	0.810	0.810	0.810	0.000	0.000
9	ImplicitFlows	0.000	0.000	0.000	0.000	0.000
10	InterAppCommunication	0.000	0.625	0.625	0.000	-0.625
11	InterComponentCommunication	0.348	0.750	0.690	0.060	-0.342
12	Lifecycle	0.769	0.933	0.769	0.164	0.000
13	Native	0.000	0.333	0.889	-0.556	-0.889
14	Reflection	0.095	0.333	0.800	-0.467	-0.705
15	Reflection_ICC	0.000	0.000	0.000	0.000	0.000
16	SelfModification	0.000	0.000	0.000	0.000	0.000
17	Threading	1.000	1.000	1.000	0.000	0.000
18	UnreachableCode	1.000	1.000	1.000	0.000	0.000
	Ø	0.504	0.632	0.646	-0.014	-0.142



Benchmarks: Present

App Merging for Benchmark Speed-Up and Analysis Lift-Up [???'19]

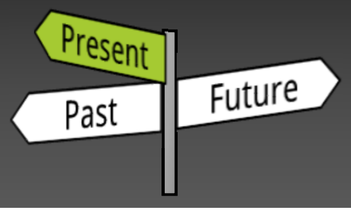
- Continuous Integration
  - Requires: Benchmark optimization
  - Approach: App merging



Benchmarks: Present

App Merging for Benchmark Speed-Up and Analysis Lift-Up [???'19]

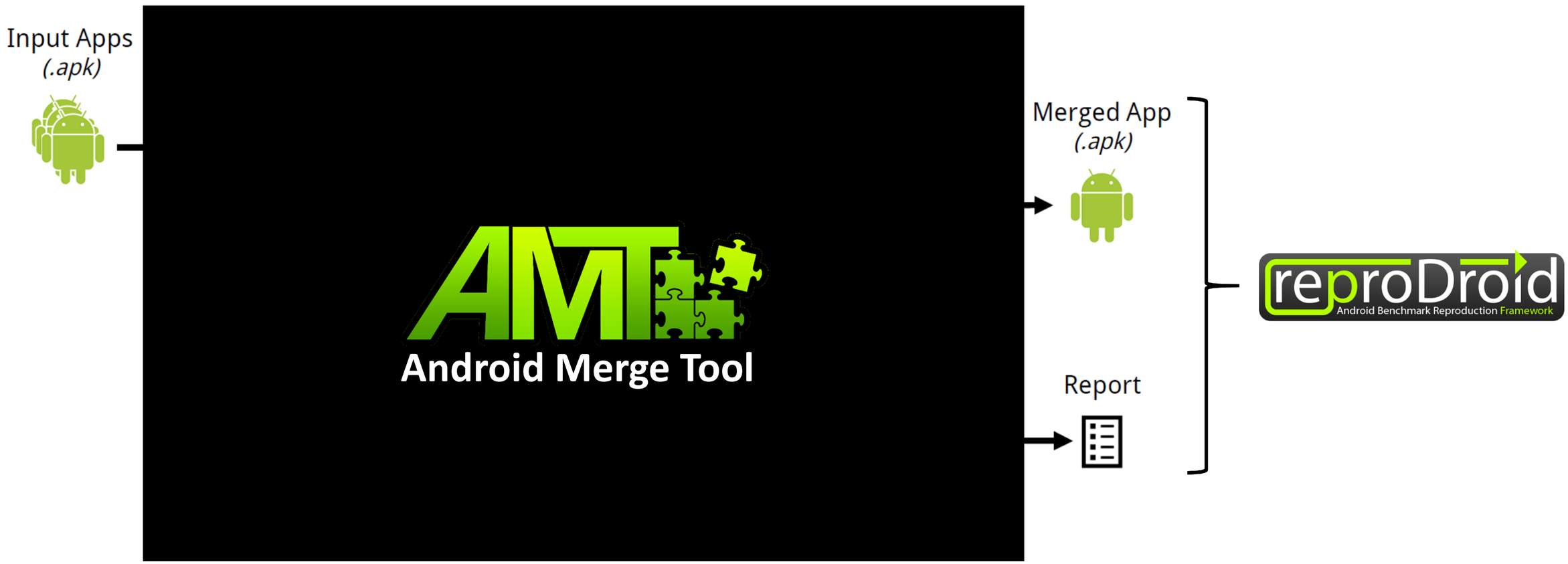
- Continuous Integration
  - Requires: Benchmark optimization
  - Approach: App merging

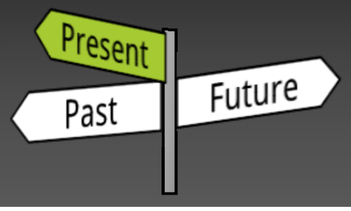


Benchmarks: Present

App Merging for Benchmark Speed-Up and Analysis Lift-Up [???'19]

- Continuous Integration
  - Requires: Benchmark optimization
  - Approach: App merging

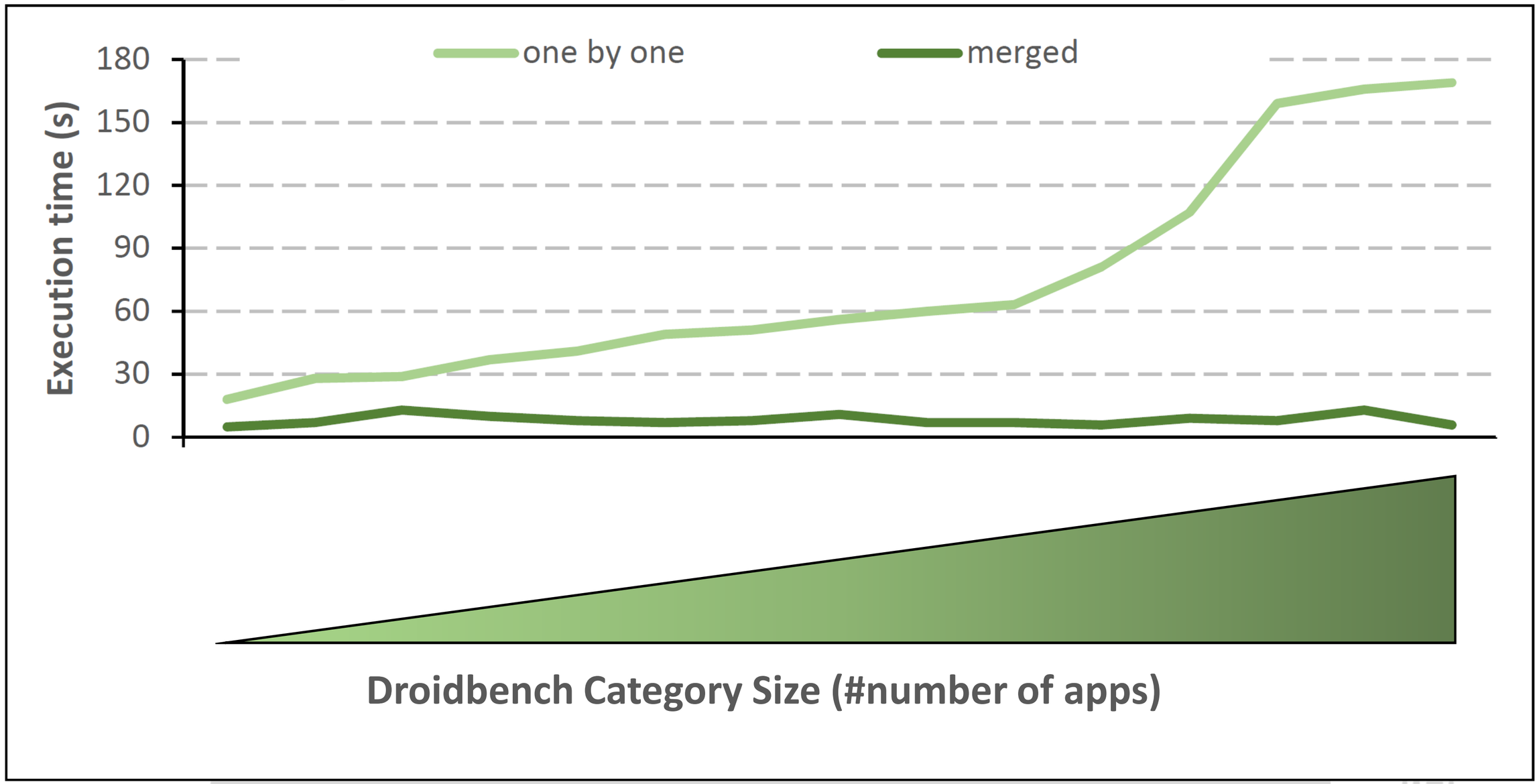


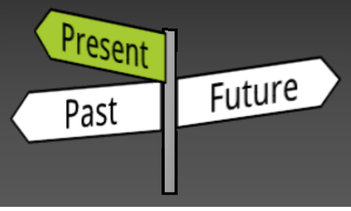


Benchmarks: Present

App Merging for Benchmark Speed-Up and Analysis Lift-Up [???'19]

- Continuous Integration

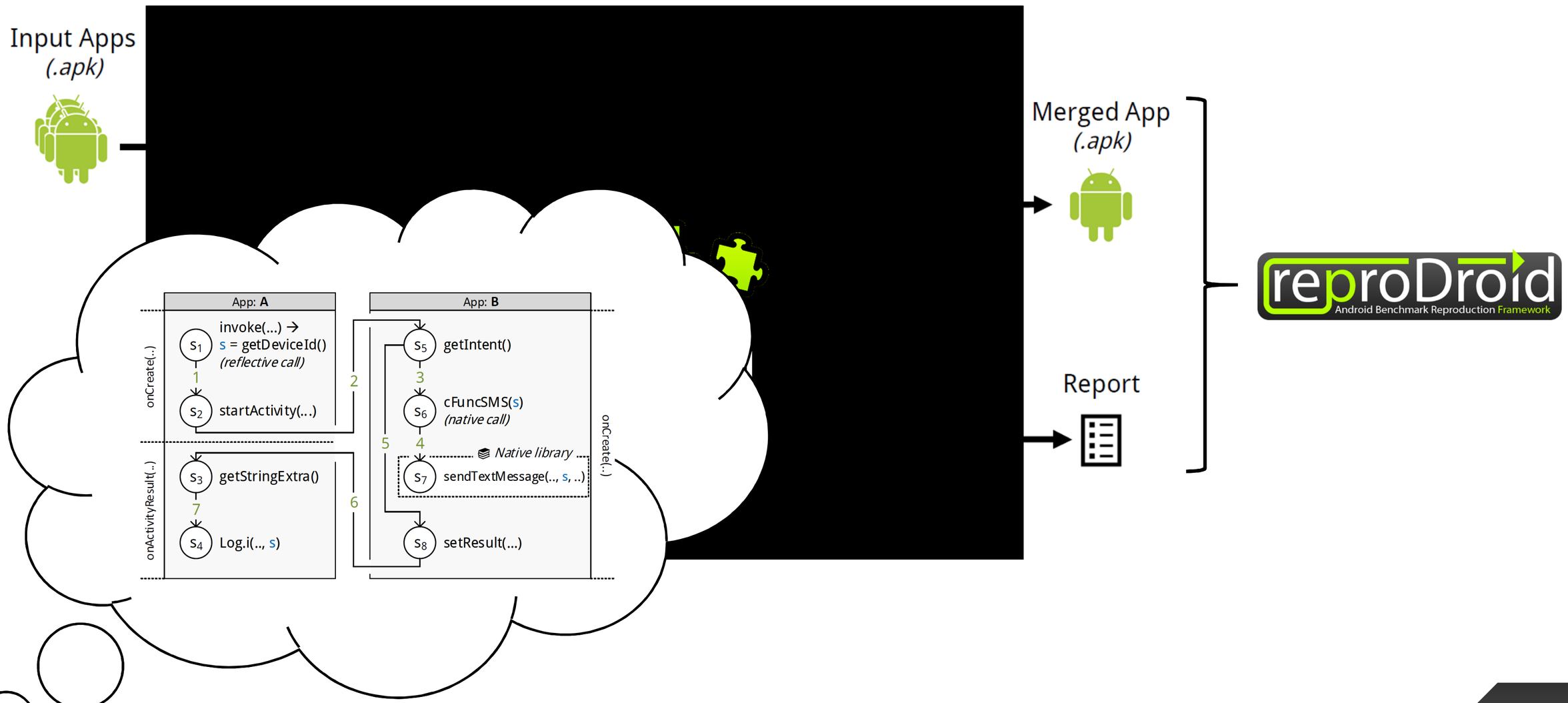


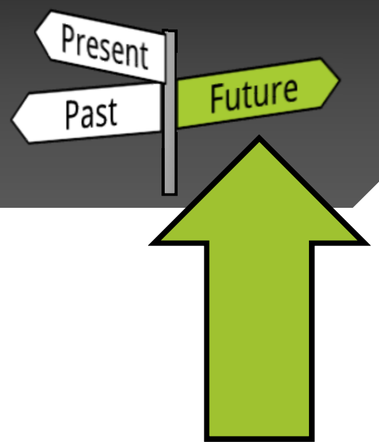


Benchmarks: Present

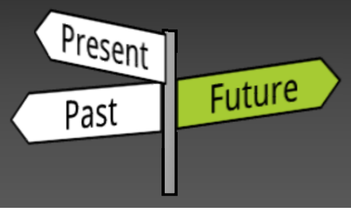
App Merging for Benchmark Speed-Up and Analysis Lift-Up [???'19]

- Continuous Integration
  - Requires: Benchmark optimization
  - Approach: App merging



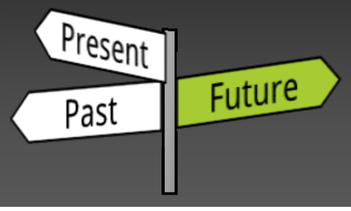


Benchmarks: Future



Benchmarks: Future

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



Benchmarks: Future

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

ISSTA 2019 Technical Papers

About | Program | Call for Submissions | Double-Blind Reviewing | Submission Policies

Accepted Papers

Accepted Papers

★ **Title**

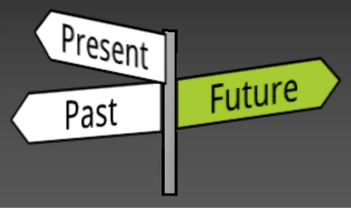
☆ A Large-Scale Study of Application Incompatibilities in Android
Haipeng Cai, Ziyi Zhang, Li Li, Xiaoqin Fu
Pre-print

☆ Adlib: Analyzer for Mobile Ad Platform Libraries
Sungho Lee, Sukyoung Ryu

☆ Assessing the State and Improving the Art of Parallel Testing for C
Oliver Schwahn, Nicolas Coppik, Stefan Winter, Neeraj Suri

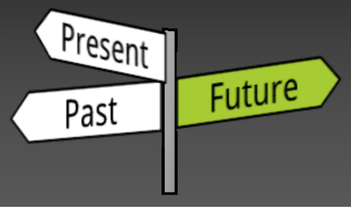
☆ Automated API-Usage Update for Android Apps
Mattia Fazzini, Qi Xin, Alessandro Orso

☆ Automatically Testing Self-Driving Cars with Search-based Procedural Content Generation
Alessio Gambi, Marc Mueller, Gordon Fraser



Benchmarks: Future

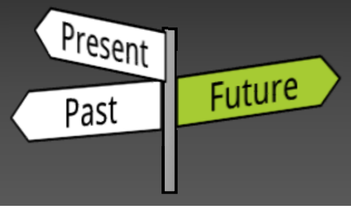
- Standardized evaluation procedure,
- Large userbase,
- A collection of up-to-date apps,
- Executable,
- Automatic evaluation,
- Reproducible.



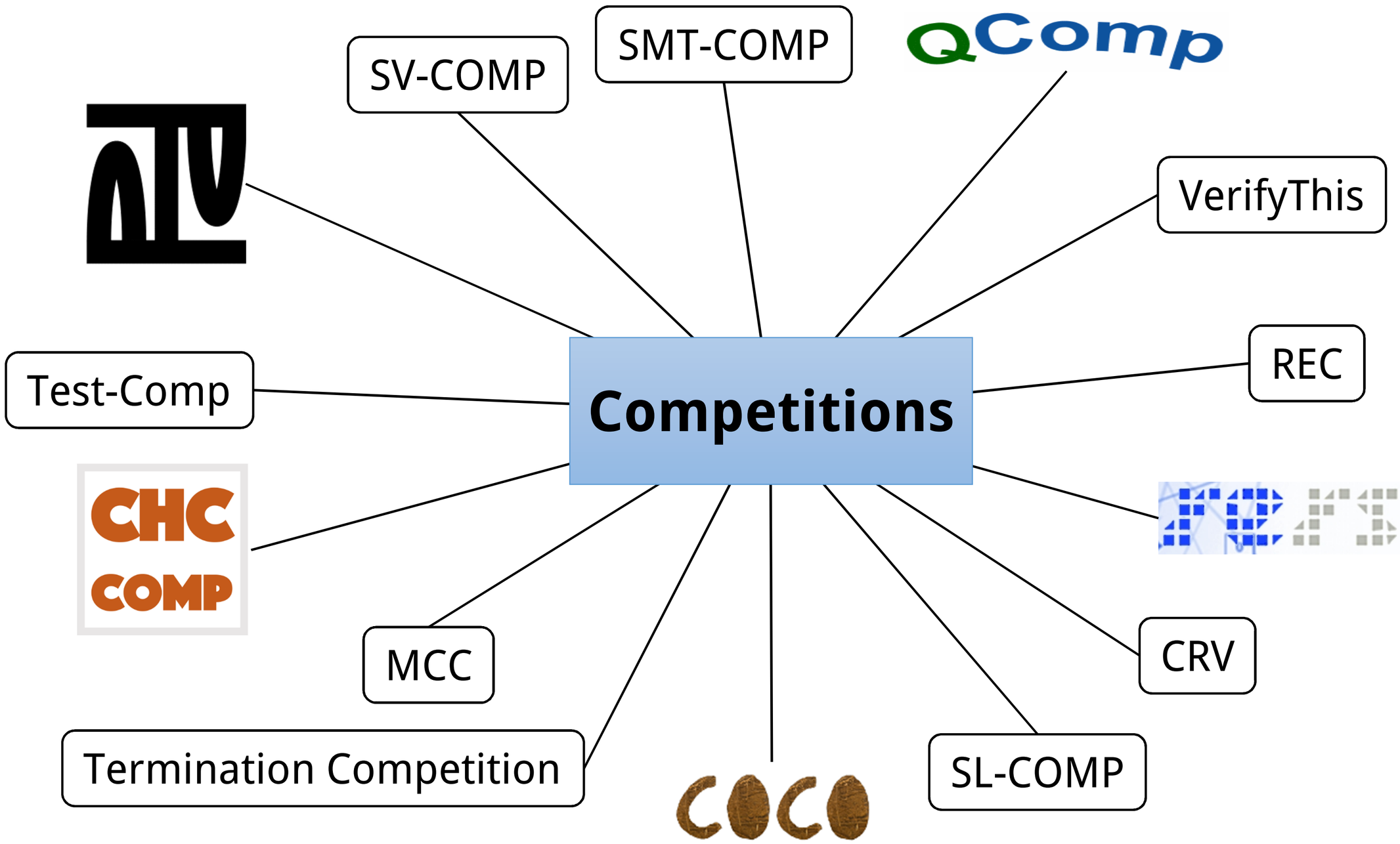
Benchmarks: Future

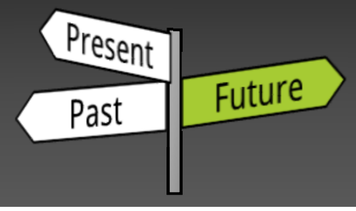
- Standardized evaluation procedure,
- Large userbase,
- A collection of up-to-date apps,
- Executable,
- Automatic evaluation,
- Reproducible.

Competitions

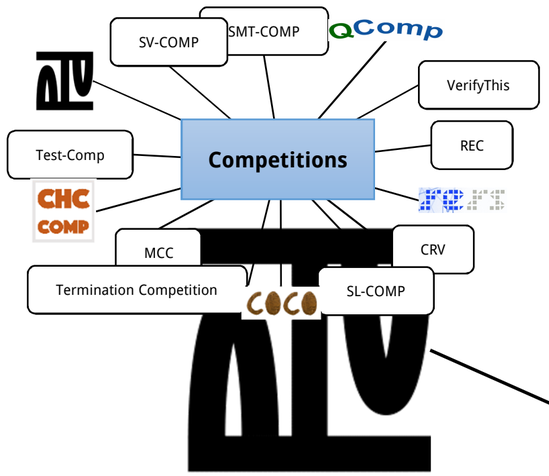


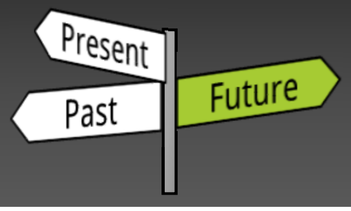
Benchmarks: Future



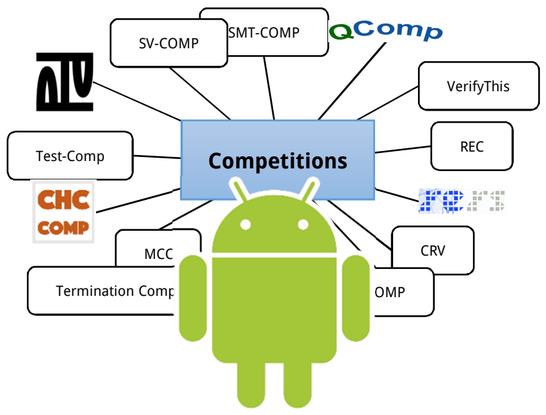


Benchmarks: Future





Benchmarks: Future

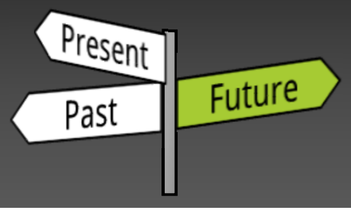


- **Challenges**

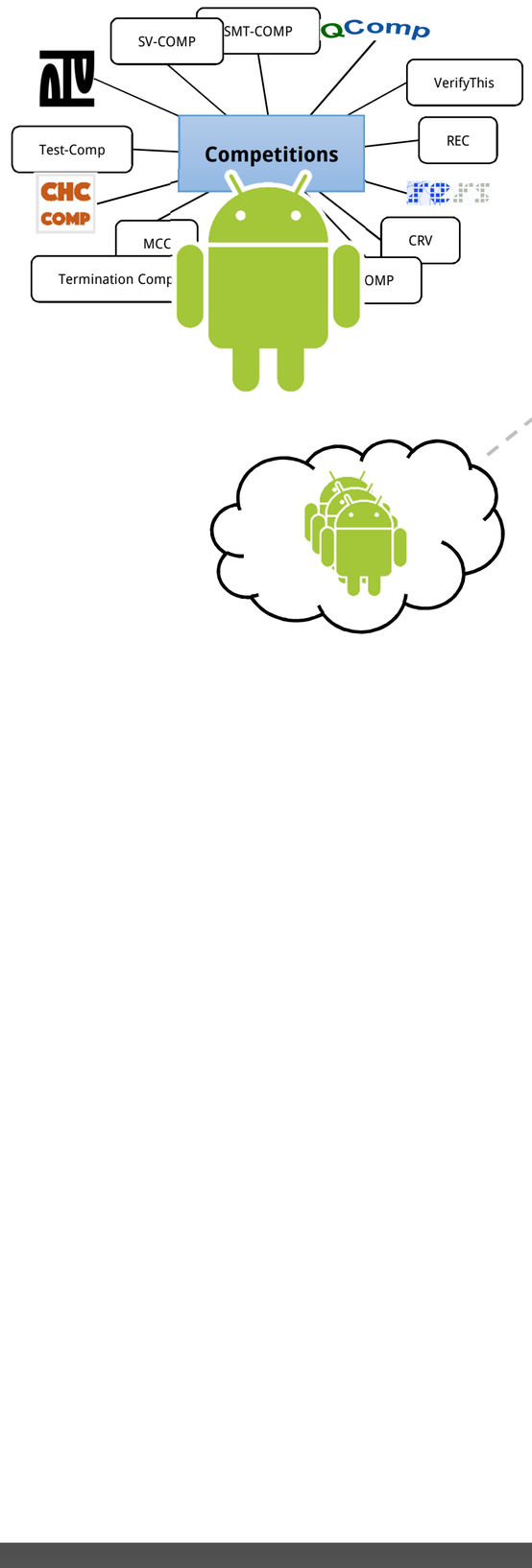
ID	Category	Author	App (.apk)	Source code (.zip)	True positives	False positives
0	Native	foellix.github.io	SourceInNative.apk	SourceInNative.zip	0_tp.xml	0_fp.xml

- **Participants**
  1. Analysis tool

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



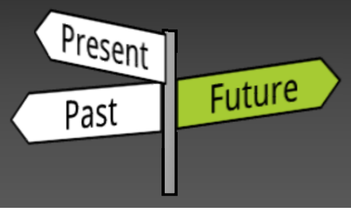
Benchmarks: Future



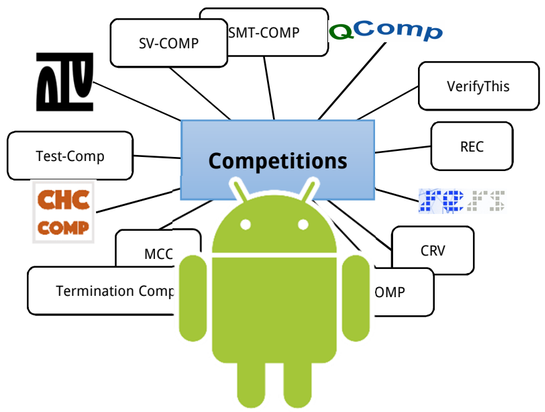
- **Challenges**

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



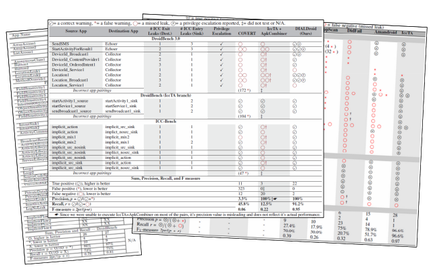
Benchmarks: Future



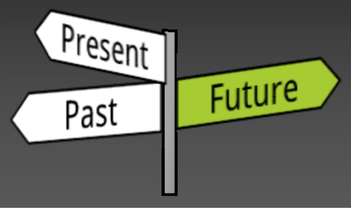
- **Challenges**

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

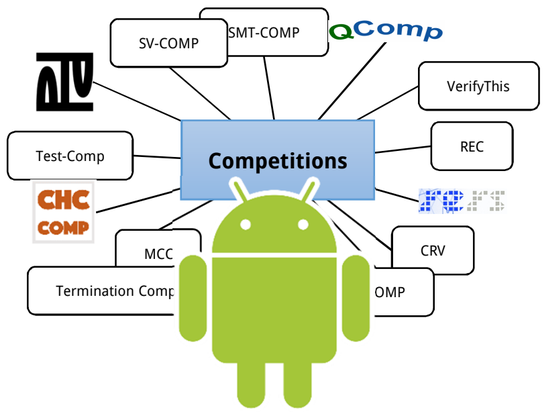
- **Participants**
  1. Analysis tool

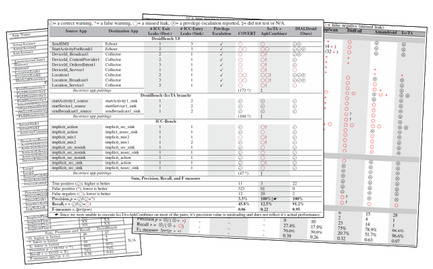


- **Committee**
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition



Benchmarks: Future



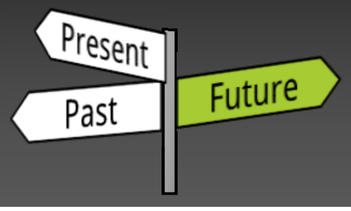


- **Challenges**

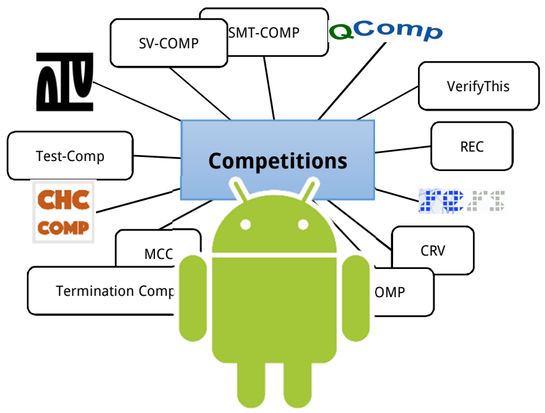
ID	Category	Author	App (.apk)	Source code (.zip)	True positives	False positives
0	Native	foellix.github.io	SourceInNative.apk	SourceInNative.zip	0_tp.xml	0_fp.xml

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

reproDroid
Android Benchmark Reproduction Framework



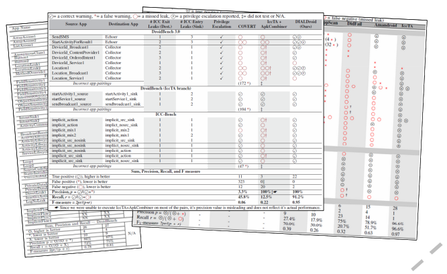
Benchmarks: Future



- **Challenges**

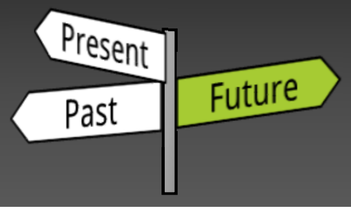
ID	Category	Author	App (.apk)	Source code (.zip)	True positives	False positives
0	Native	foellix.github.io	SourceInNative.apk	SourceInNative.zip	0_tp.xml	0_fp.xml

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

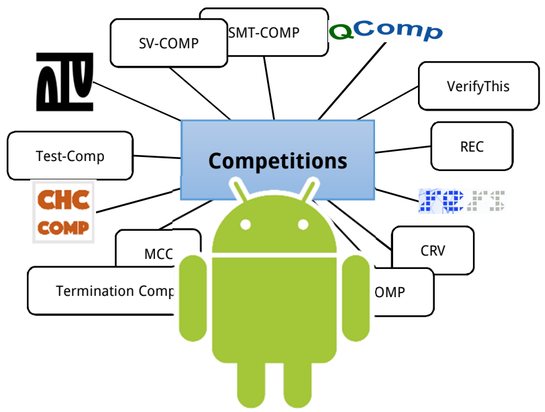


- **Committee**
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition

reproDroid
Android Benchmark Reproduction Framework

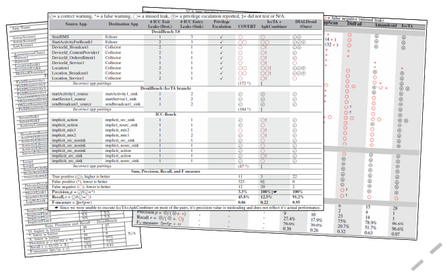


Benchmarks: Future



- **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

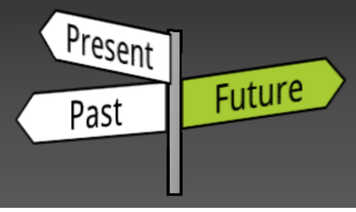


- **Participants**
  1. Analysis tool
  2. *Configuration*
  3. *Converter*

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

reproDroid
Android Benchmark Reproduction Framework

When do we start an Android Taint-Analysis Competition?



Benchmarks: Future

- **Challenges**

ID	Category	Author	False positives
0	Native	foellix.github	fp.xml

- **Participant**
  1. Analy
  2. *Config*
  3. *Conve*

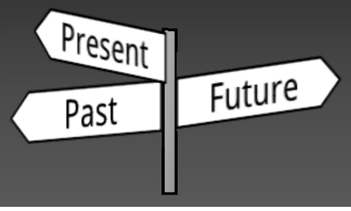
- **Committee**
  1. Review:
     - Challenge submissions
     - Tool submissions (+ associated tool-papers)
  2. Run the competition

reproDroid
Android Benchmark Reproduction Framework

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

Organization, Venue, Prices, Proceedings (Competition report), … ?

When do we start an Android Taint-Analysis Competition?



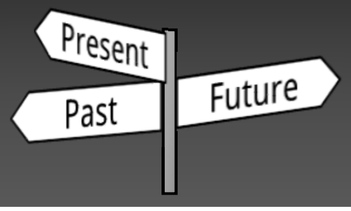
Summary

Android Taint-Analysis Benchmarks: Got, Get and will get better!

Past 

- ✚ Eva. procedure,
- ✚ Userbase,
- ▬ Up-to-date,
- ▬ Executability,
- ▬ Automation,
- ▬ Reproducibility.

Github: https://FoelliX.github.io



Summary

Android Taint-Analysis Benchmarks: Got, Get and will get better!

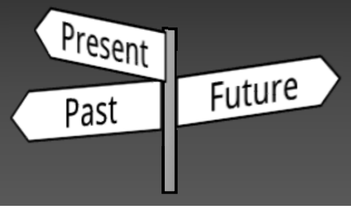
Past 

- ✚ Eva. procedure,
- ✚ Userbase,
- ▬ Up-to-date,
- ▬ Executability,
- ▬ Automation,
- ▬ Reproducibility.

Present 

- ✚ Executability,
- ✚ Automation,
- ✚ Reproducibility.

Github: https://FoelliX.github.io



Summary

Android Taint-Analysis Benchmarks: Got, Get and will get better!

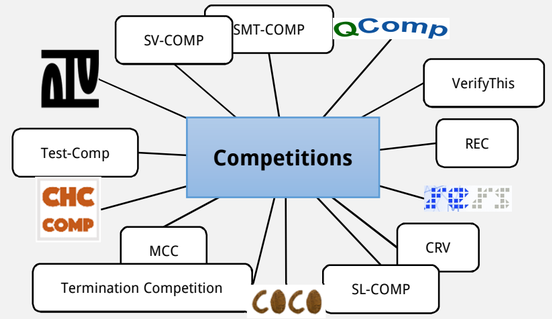
Past

- ✚ Eva. procedure,
- ✚ Userbase,
- ▬ Up-to-date,
- ▬ Executability,
- ▬ Automation,
- ▬ Reproducibility.

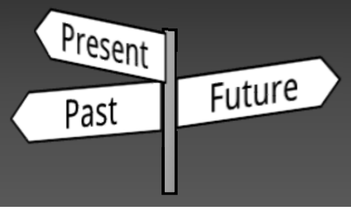
Present

Future

- ✚ Up-to-date,
- ✚ Executability,
- ✚ Automation,
- ✚ Reproducibility.



Github: https://FoelliX.github.io



Summary

Android Taint-Analysis Benchmarks: Got, Get and will get better!

Past

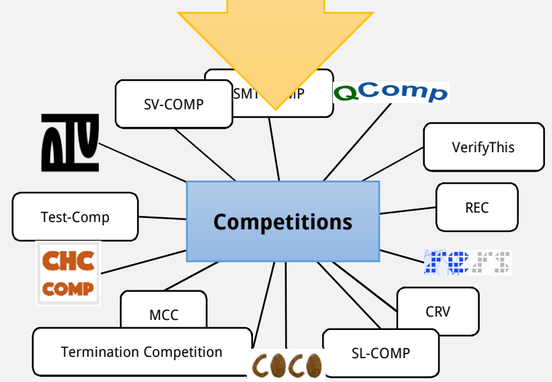
- ✚ Eva. procedure,
- ✚ Userbase,
- ▬ Up-to-date,
- ▬ Executability,
- ▬ Automation,
- ▬ Reproducibility.

Present

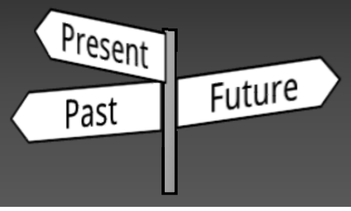
Future

- ✚ Up-to-date,
- ✚ Executability,
- ✚ Automation,
- ✚ Reproducibility.

Workshop:
Android Taint-Analysis Competitions



Github: https://FoelliX.github.io



Example (1/3)

App: **A**

App: **B**

onCreate(..)

onActivityResult(..)

onCreate(..)

s_1 invoke(...) →
s = getDeviceId()
(reflective call)

1

s_2 startActivity(...)

2

s_3 getStringExtra()

7

s_4 Log.i(.., s)

6

s_5 getIntent()

3

s_6 cFuncSMS(s)
(native call)

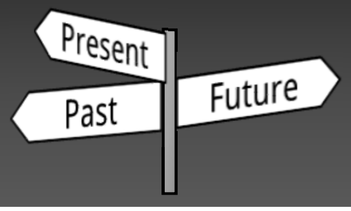
5

4

Native library

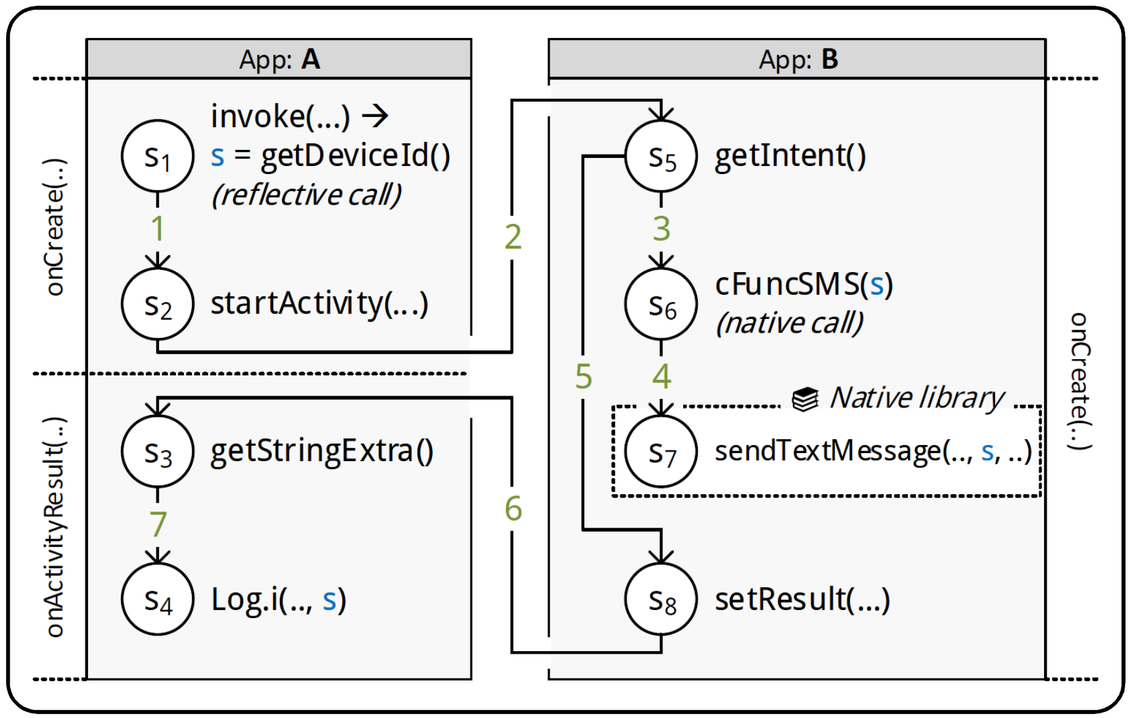
s_7 sendTextMessage(.., s, ..)

s_8 setResult(...)



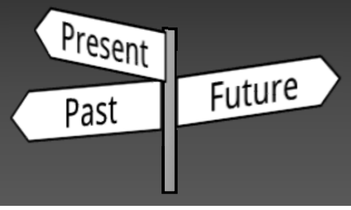
Example (2/3)

-Query



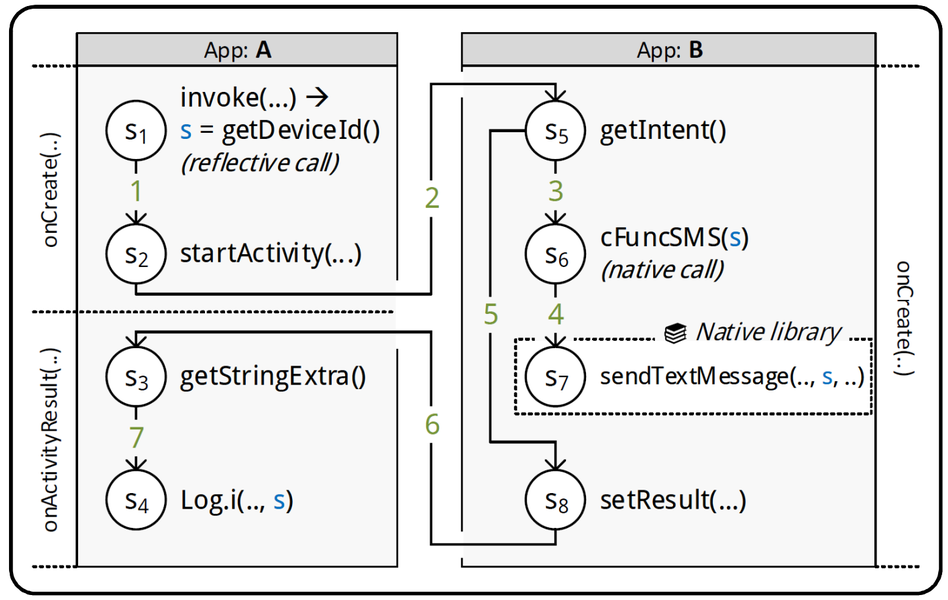

```
Flows FROM App('A.apk') TO App('B.apk') ?
```

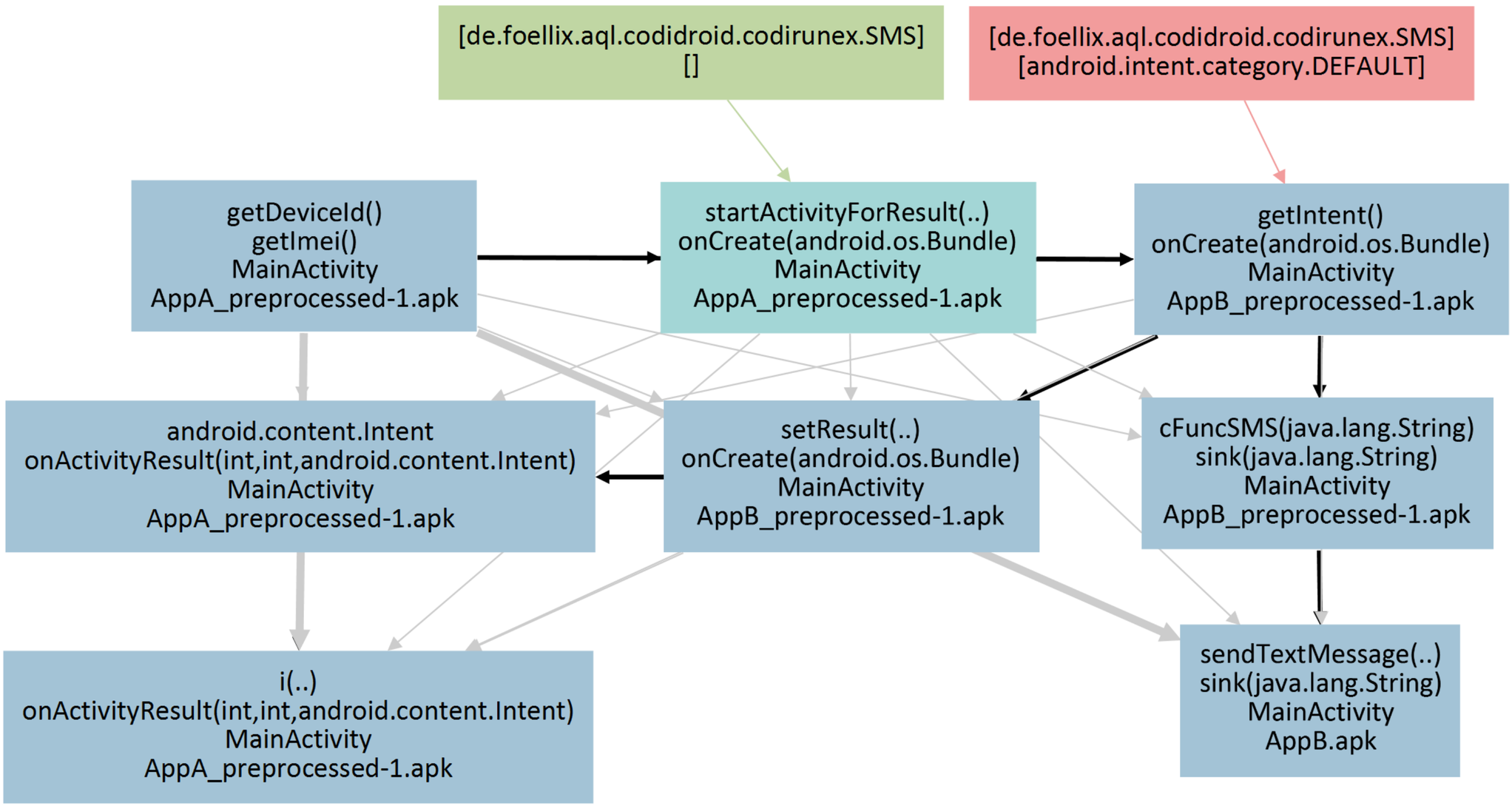
```
MATCH [
     Flows IN App('A.apk' | 'DEOBFUSCATE') ?,
     CONNECT [
          Flows IN App('B.apk' | 'UNCOVER') ?,
          Flows IN App('B.apk' | 'UNCOVER') FEATURING 'NATIVE' ?
     ],
     IntentSources IN App('A.apk' | 'DEOBFUSCATE') ?,
     IntentSinks IN App('A.apk' | 'DEOBFUSCATE') ?,
 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

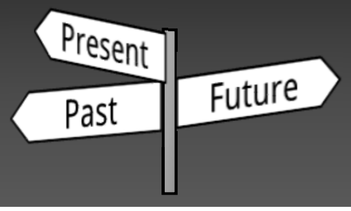


Example (3/3)

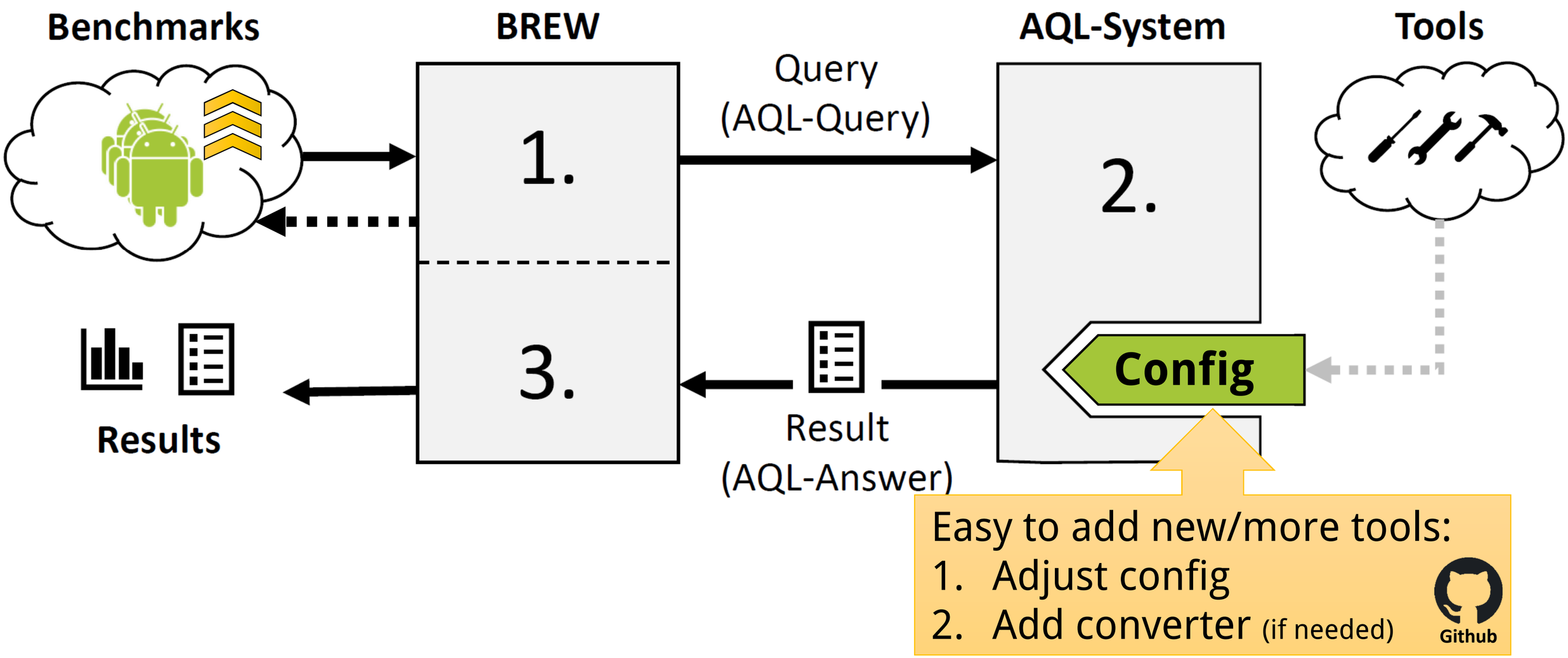
-Answer



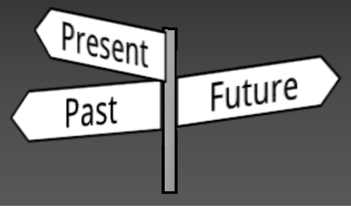




ReproDroid



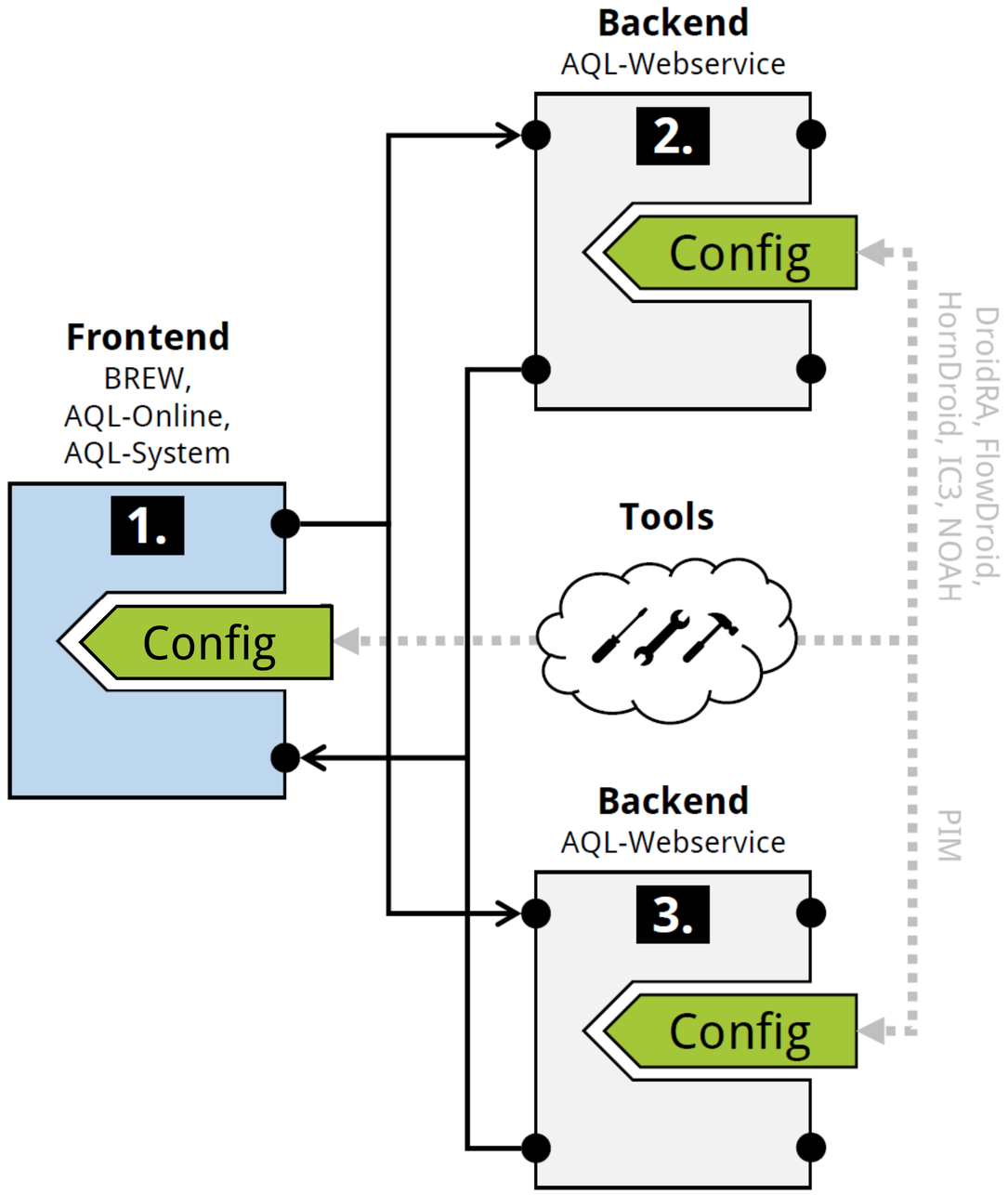
1. **Refine:** Colloquial ground-truth → Machine readable (semi-automatic)
2. **Execute:** Run arbitrary tools (automatic)
3. **Collect & Evaluate:** Precision, Recall, F-measure (automatic)

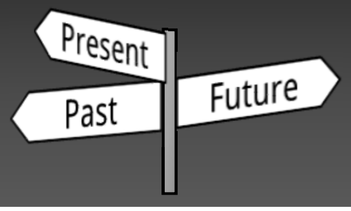


CoDiDroid

CoDiDroid
together strong!

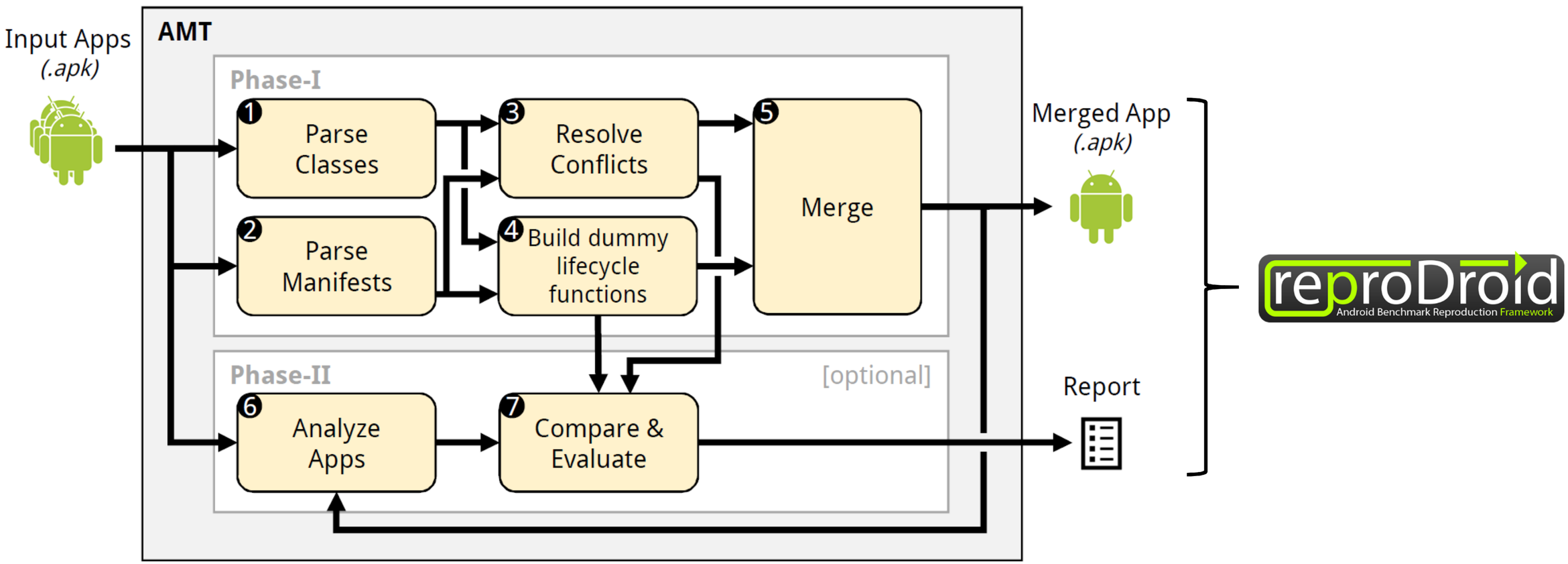
ID	Category	FlowDroid	Best	CoDiDroid	Difference to Best	Difference to FlowDroid
1	Aliasing	0.667	0.667	0.667	0.000	0.000
2	AndroidSpecific	0.900	0.900	0.900	0.000	0.000
3	ArraysAndLists	0.615	0.667	0.615	0.052	0.000
4	Callbacks	0.897	0.897	0.897	0.000	0.000
5	DynamicLoading	0.000	0.500	0.000	0.500	0.000
6	EmulatorDetection	0.966	0.966	0.966	0.000	0.000
7	FieldAndObjectSensitivity	1.000	1.000	1.000	0.000	0.000
8	GeneralJava	0.810	0.810	0.810	0.000	0.000
9	ImplicitFlows	0.000	0.000	0.000	0.000	0.000
10	InterAppCommunication	0.000	0.625	0.625	0.000	-0.625
11	InterComponentCommunication	0.348	0.750	0.690	0.060	-0.342
12	Lifecycle	0.769	0.933	0.769	0.164	0.000
13	Native	0.000	0.333	0.889	-0.556	-0.889
14	Reflection	0.095	0.333	0.800	-0.467	-0.705
15	Reflection_ICC	0.000	0.000	0.000	0.000	0.000
16	SelfModification	0.000	0.000	0.000	0.000	0.000
17	Threading	1.000	1.000	1.000	0.000	0.000
18	UnreachableCode	1.000	1.000	1.000	0.000	0.000
	Ø	0.504	0.632	0.646	-0.014	-0.142

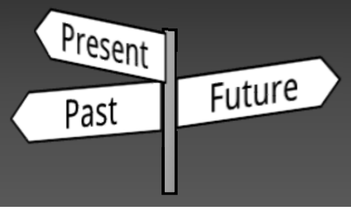



Present
Past
Future

AMT


AMT

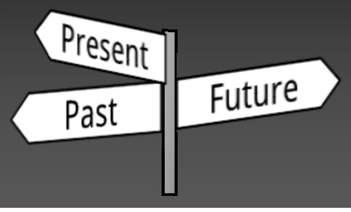

Input Apps
(.apk)
AMT
Phase-I
1 Parse Classes
2 Parse Manifests
3 Resolve Conflicts
4 Build dummy lifecycle functions
5 Merge
Phase-II
[optional]
6 Analyze Apps
7 Compare & Evaluate
Merged App
(.apk)
Report
reproDroid
Android Benchmark Reproduction Framework



AQL: Queries

```
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')
?
```

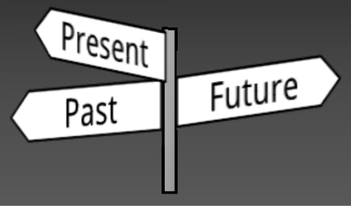


AQL: Answers

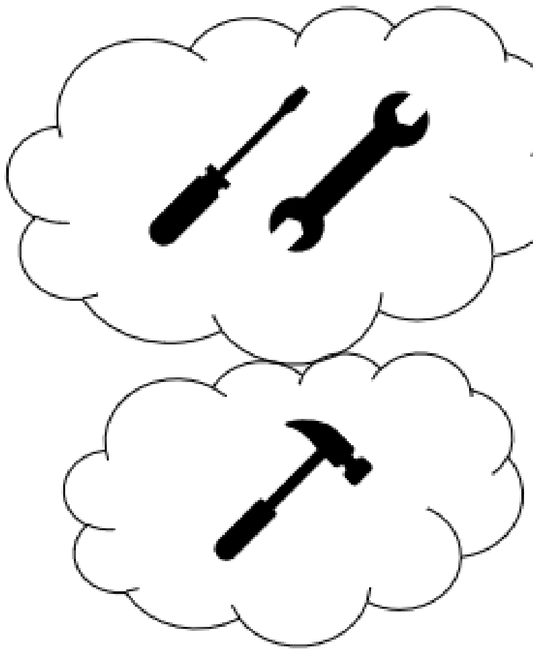
```
Found a flow to sink virtualinvoke $r4.<android.telephony.
    → SmsManager: void sendTextMessage(java.lang.String,
    → java.lang.String,java.lang.String,android.app.
    → PendingIntent,android.app.PendingIntent)>("+49 1234",
    →  null, $r5, null, null), from the following sources:
      - $r5 = 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,
    → java.lang.String,android.app.PendingIntent,android.
    → app.PendingIntent)>': ###
['Src: <android.telephony.TelephonyManager: java.lang.String
    →  getDeviceId()>']
```

```
<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>
```

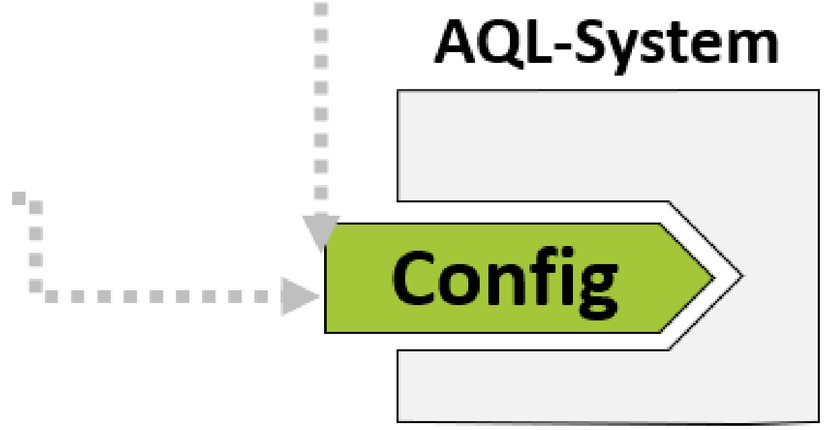


AQL: Add a new/more tools

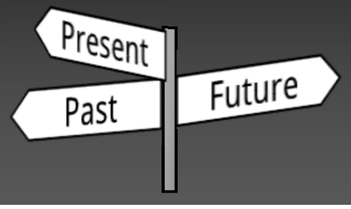


- Analysis tools, Preprocessors, Operators
- Converter

new



- **~ 140 LOC for our 6 converters**


Present
Past
Future