Call Graph Soundness in Android Static Analysis

Jordan Samhi[1], René Just[2], Tegawendé F. Bissyandé[3], Michael D. Ernst[2], Jacques Klein[3]

[1]*CISPA Helmholtz Center for Information Security*

[2]*University of Washington*

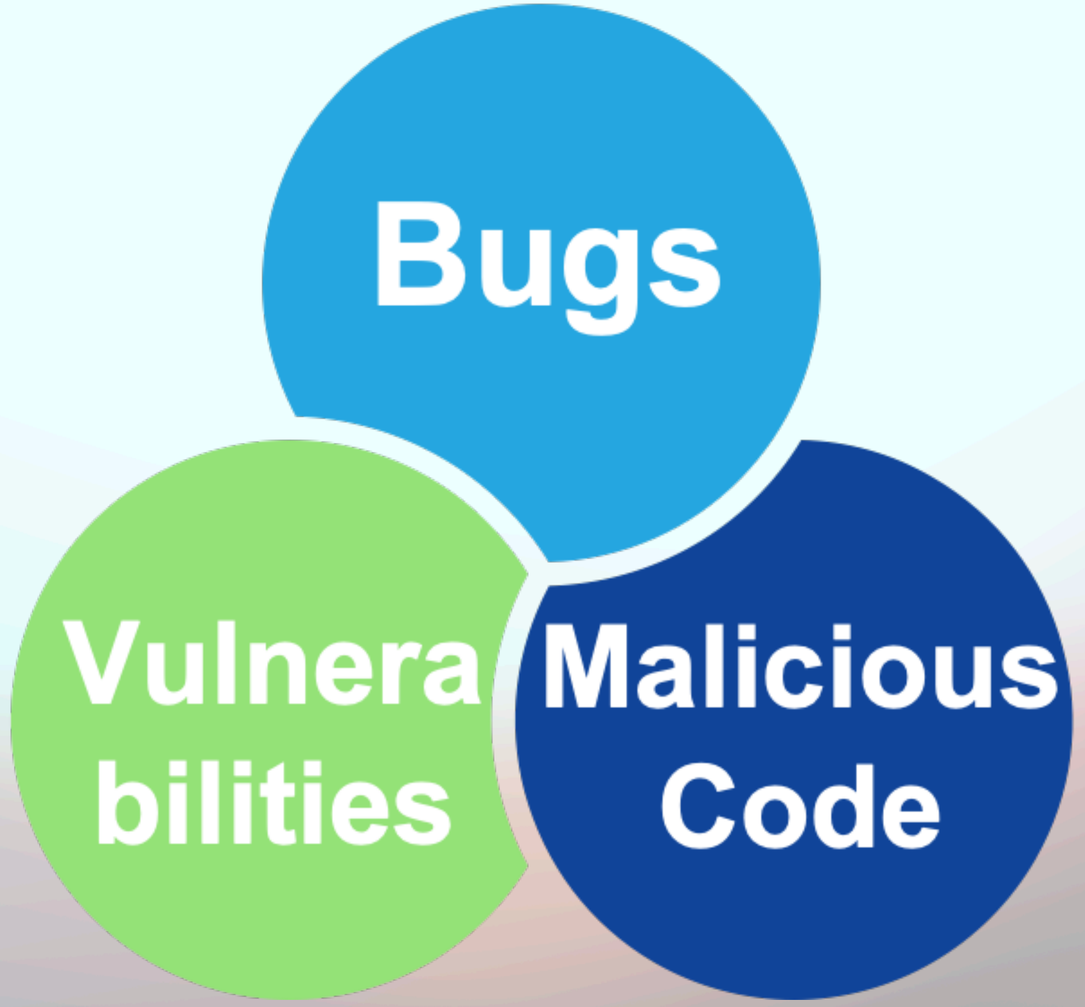
[3]*University of Luxembourg*

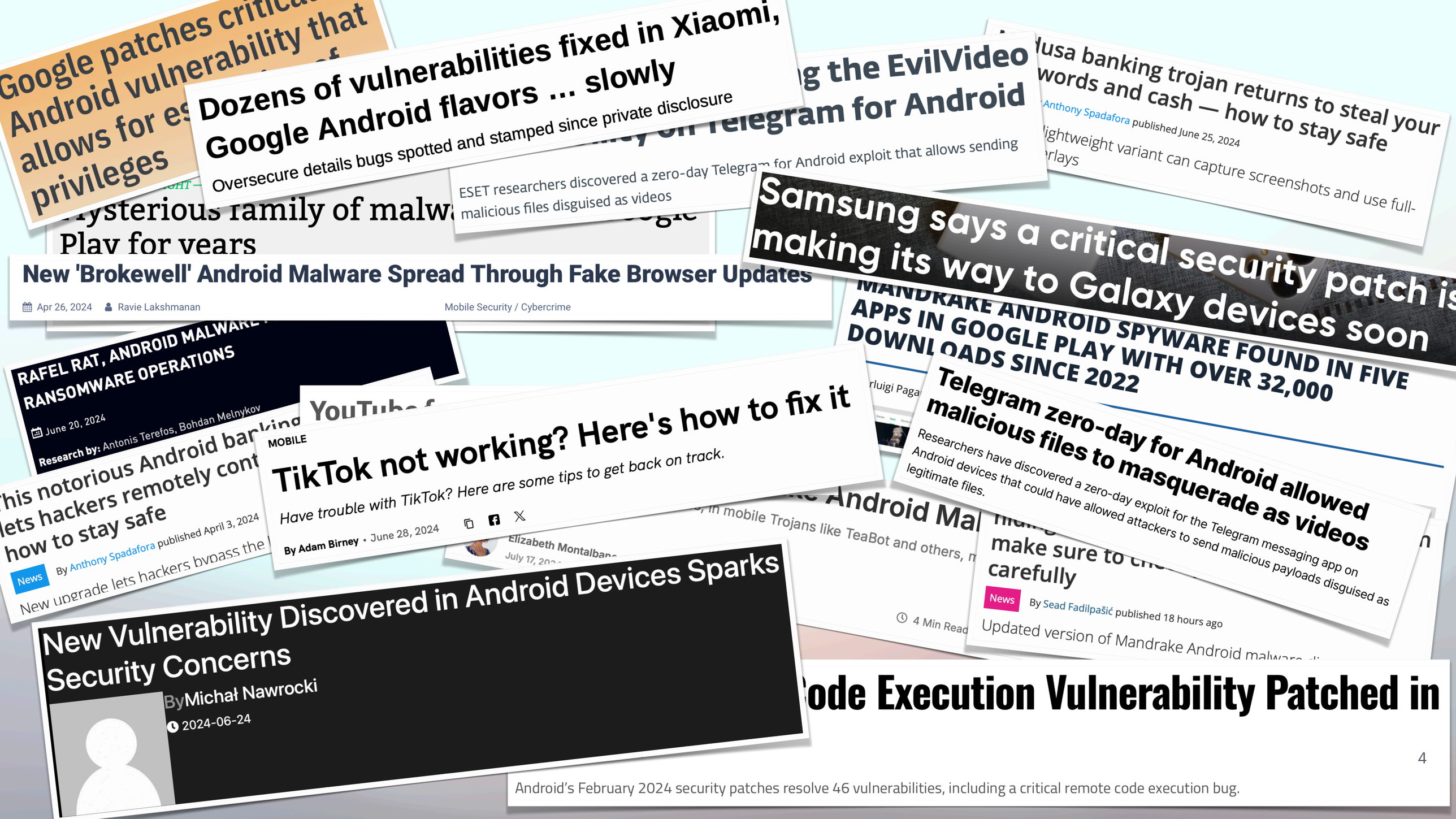
September, 20th 2024 - ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)

Basic statistics

- More than 6 billion people own a smartphone
- Almost three-quarters are Android-based
- We manipulate a lot of sensitive data

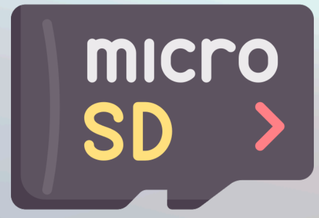
High Security Risks

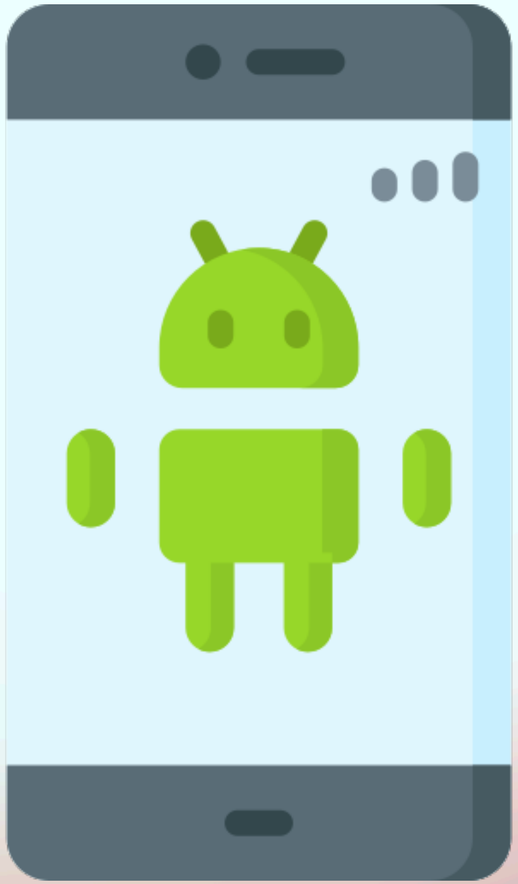



Google patches critical Android vulnerability that allows for es privileges
Dozens of vulnerabilities fixed in Xiaomi, Google Android flavors … slowly
Oversecure details bugs spotted and stamped since private disclosure
g the EvilVideo Telegram for Android
ESET researchers discovered a zero-day Telegram for Android exploit that allows sending malicious files disguised as videos
lusa banking trojan returns to steal your words and cash — how to stay safe
Anthony Spadafora published June 25, 2024
ightweight variant can capture screenshots and use full-erlays
Mysterious family of malwa Play for years
New 'Brokewell' Android Malware Spread Through Fake Browser Updates
Apr 26, 2024
Ravie Lakshmanan
Mobile Security / Cybercrime
Samsung says a critical security patch is making its way to Galaxy devices soon
MANDRAKE ANDROID SPYWARE FOUND IN FIVE APPS IN GOOGLE PLAY WITH OVER 32,000 DOWNLOADS SINCE 2022
RAFEL RAT, ANDROID MALWARE RANSOMWARE OPERATIONS
June 20, 2024
Research by: Antonis Terefos, Bohdan Melnykov
his notorious Android bank lets hackers remotely cont how to stay safe
By Anthony Spadafora published April 3, 2024
News
New upgrade lets hackers bypass the
YouTube
MOBILE
TikTok not working? Here's how to fix it
Have trouble with TikTok? Here are some tips to get back on track.
By Adam Birney • June 28, 2024
Elizabeth Montalbano
July 17, 202
Telegram zero-day for Android allowed malicious files to masquerade as videos
Researchers have discovered a zero-day exploit for the Telegram messaging app on Android devices that could have allowed attackers to send malicious payloads disguised as legitimate files.
make sure to th carefully
News
By Sead Fadilpašić published 18 hours ago
Updated version of Mandrake Android malware
Android Mal
, in mobile Trojans like TeaBot and others,
4 Min Read
New Vulnerability Discovered in Android Devices Sparks Security Concerns
By Michał Nawrocki
2024-06-24
ode Execution Vulnerability Patched in
Android's February 2024 security patches resolve 46 vulnerabilities, including a critical remote code execution bug.


www


SMS


micro
SD



How to detect?

1

Dynamic Analysis

2

Static Analysis

Dynamic Analysis

Dynamic analysis is precise!

*Ernst, Michael D. "Static and dynamic analysis: Synergy and duality." WODA 2003: ICSE Workshop on Dynamic Analysis. 2003

Static Analysis

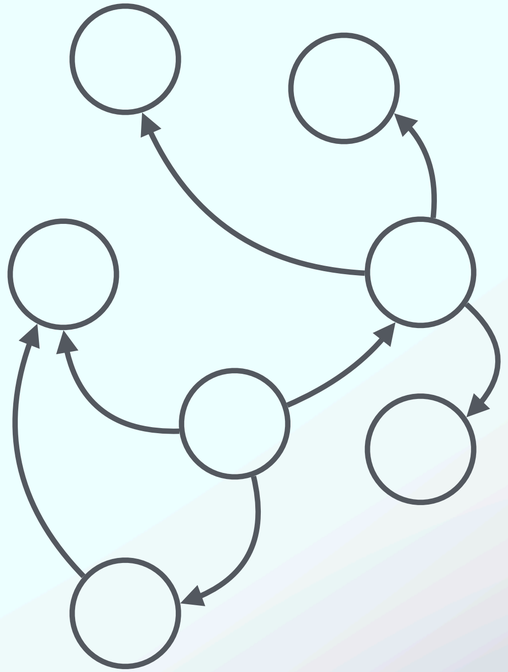
Static analysis is sound!

"Soundness guarantees that analysis results are an **accurate**

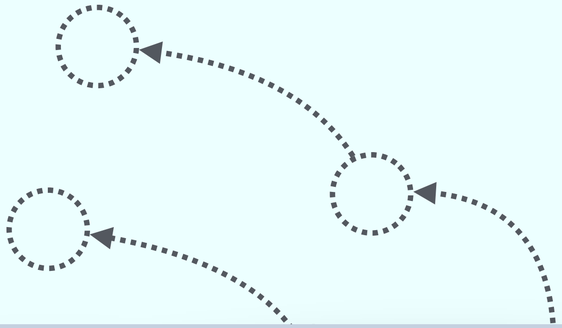
Is it?

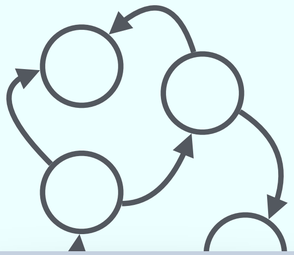
*Ernst, Michael D. "Static and dynamic analysis: Synergy and duality." WODA 2003: ICSE Workshop on Dynamic Analysis. 2003

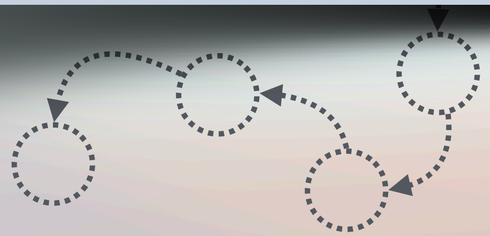
FlowDroid



- malware detection
- features extraction
- instrumentation
- incompatibility issues
- Type-state issues
- etc.







?

?

?

Can you trust this model?

ICC

- Li, Li et al. Iccta: Detecting inter-component privacy leaks in android apps. ICSE 2015.
- Wei et al., Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps. *TOPS 2018*.
- Gordon et al. Information flow analysis of android applications in droidsafe. NDSS 2015.

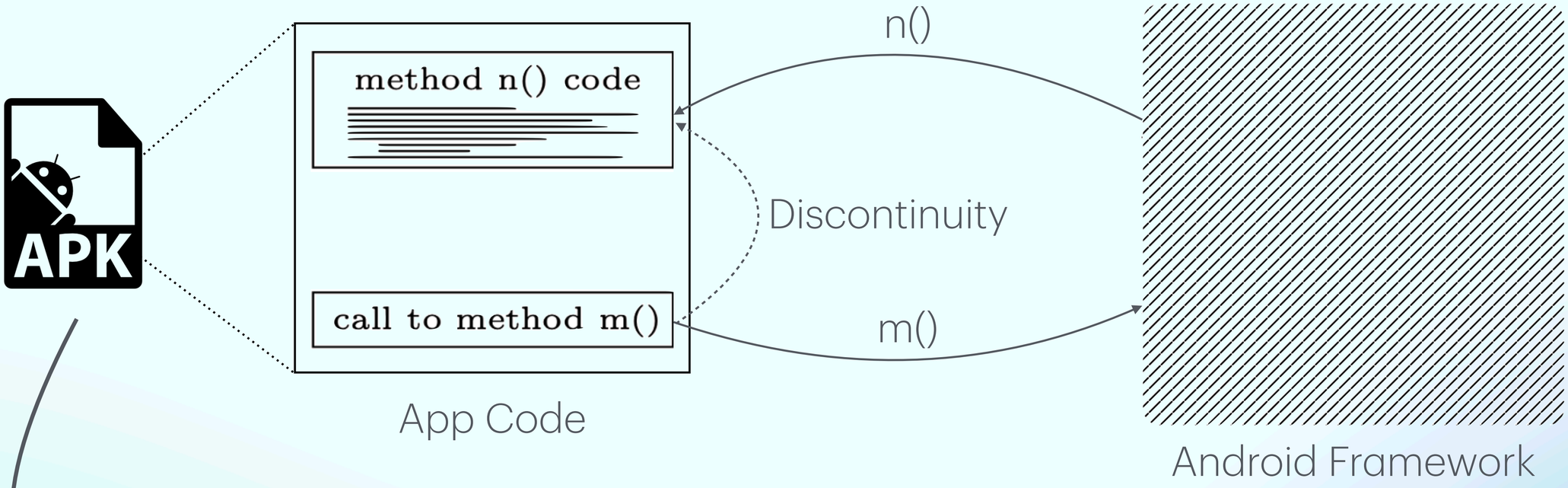
ming reflection to
analysis of android

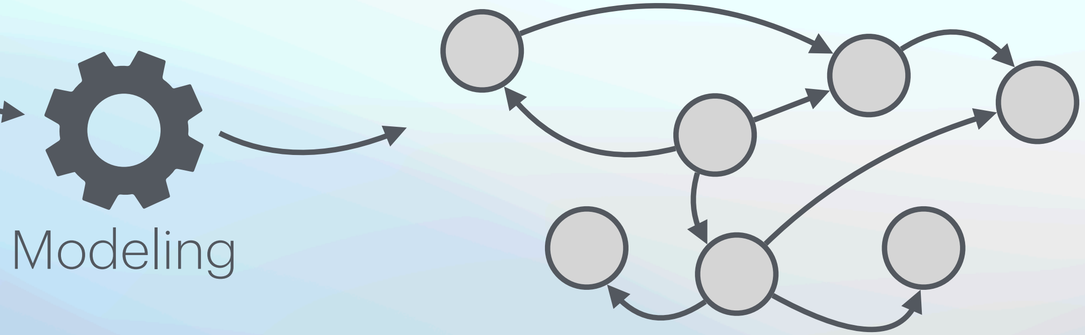
lysis of implicit control
ection and android

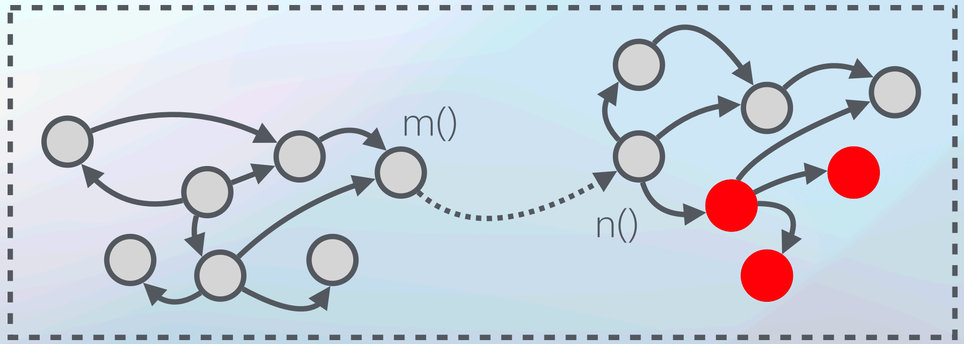
Callback

- Arzt et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. PLDI 2014.
- Yang et al. Static control-flow analysis of user-driven callbacks in Android applications. ICSE 2015.

Comprehensiveness of Program Analysis







Example

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```

Example

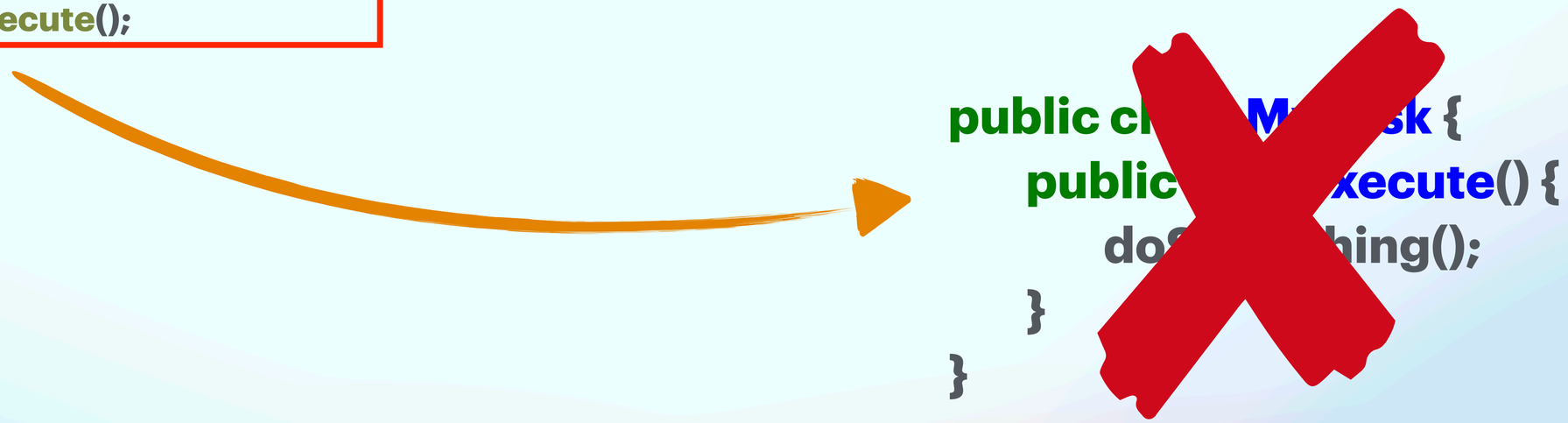
```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```

Example

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```

Example

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```



Example

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```

```
public class MyTask extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... params) {
        return "Background task completed";
    }

    @Override
    protected void onPostExecute(String result) {
        textView.setText(result);
    }
}
```

Example

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        MyTask myTask = new MyTask();
        myTask.execute();
    }
}
```

```
public class MyTask extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... params) {
        return "Background task completed";
    }

    @Override
    protected void onPostExecute(String result) {
        textView.setText(result);
    }
}
```

Example

execute ▪▪▪▪ **doInBackground** ▪▪▪▪ **onPostExecute**

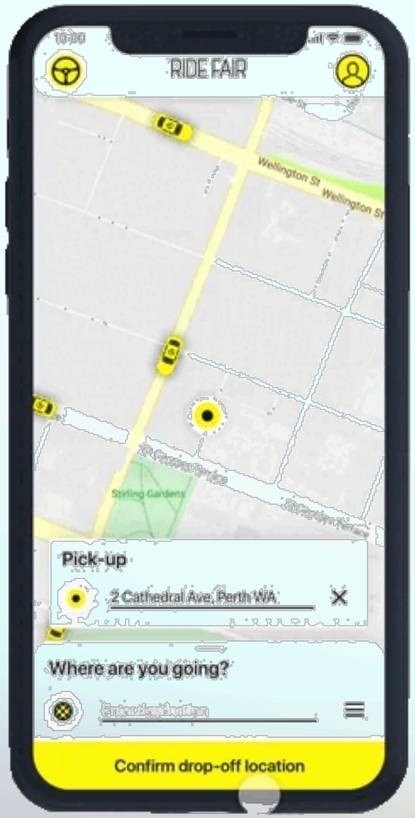
If static analysis tools do not model these discontinuities, the model is

unsound

Objective

Measure and understand the level of **unsoundness** in Android static analysis tools

How?



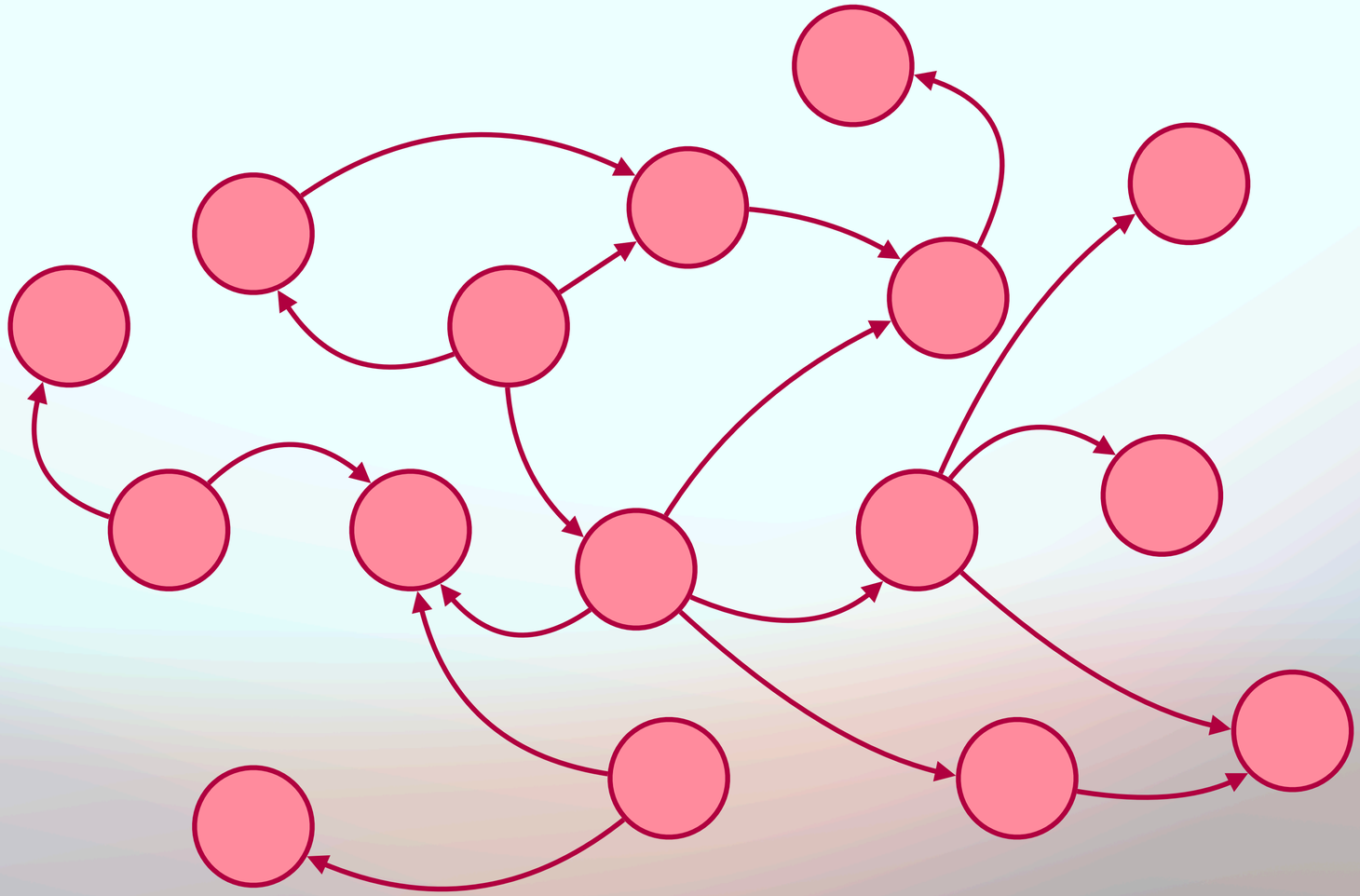
Dynamic Analysis

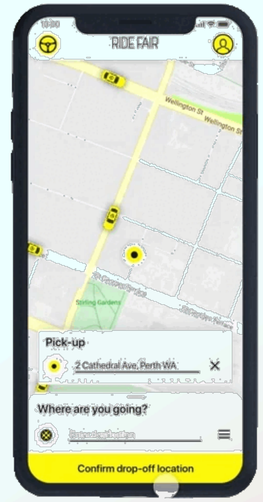
Static Analysis

Dataset

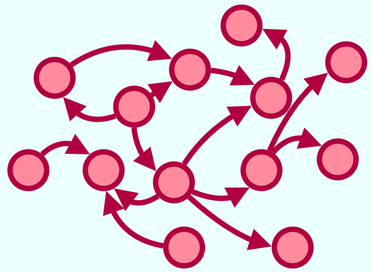
1000 apps from AndroZoo from 2023

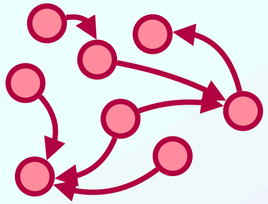
Dynamic Analysis

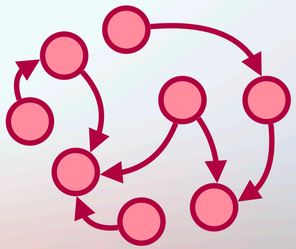


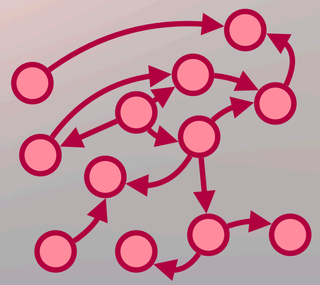


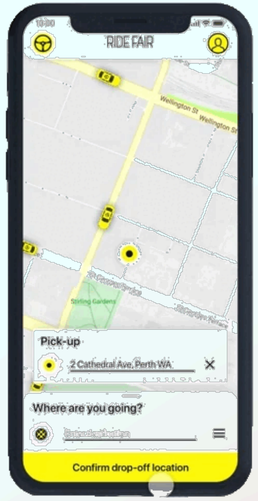
Dynamic Analysis



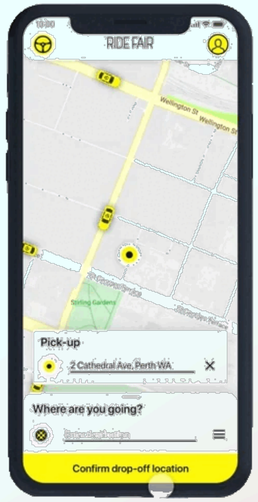






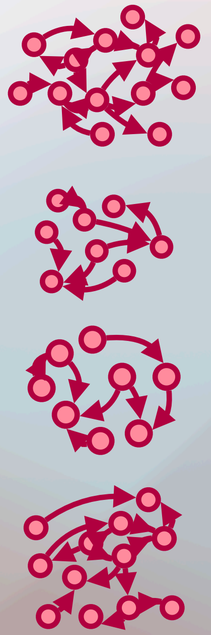


Dynamic Analysis



1000

call graphs



8%

Average Code Coverage

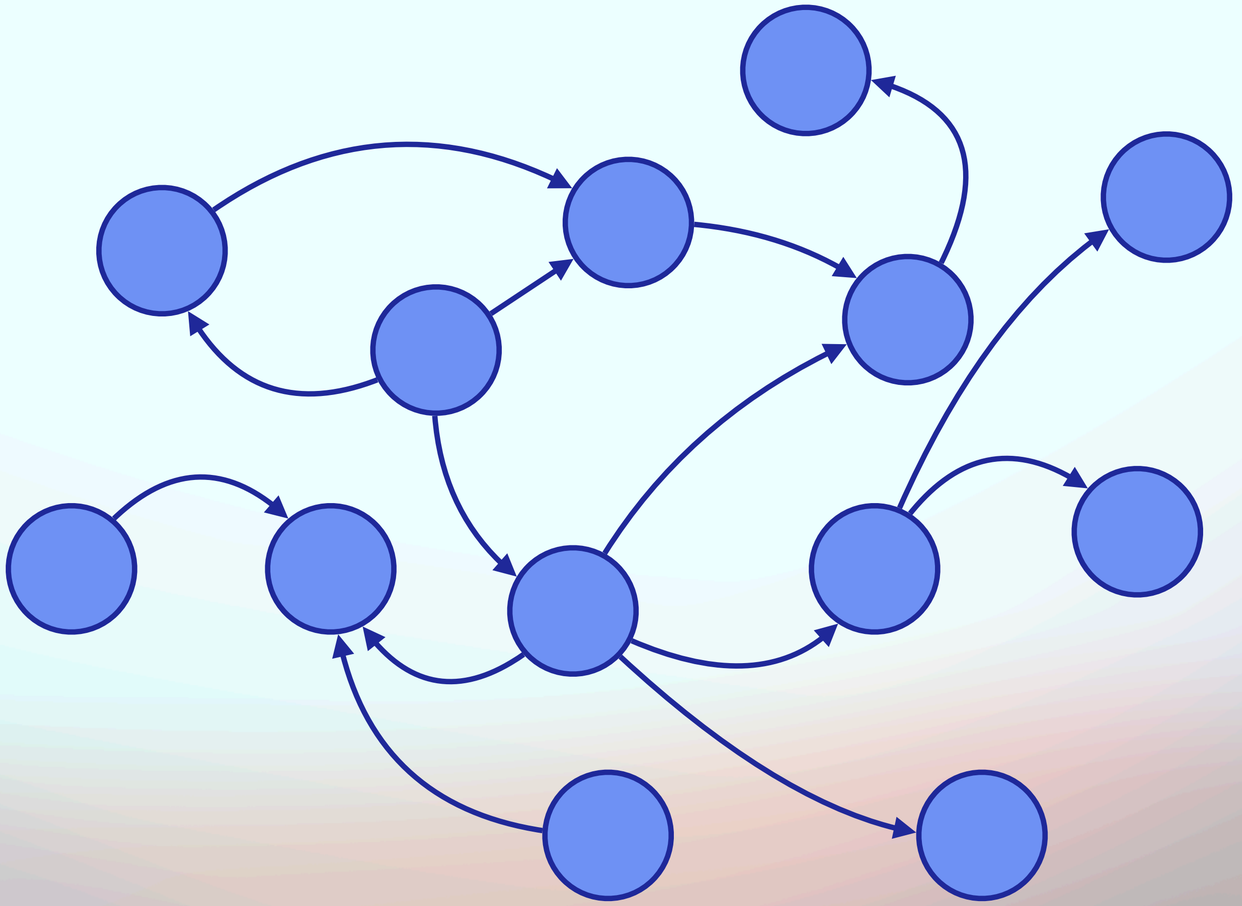
8%

Average Code Coverage

4%

Median Code Coverage

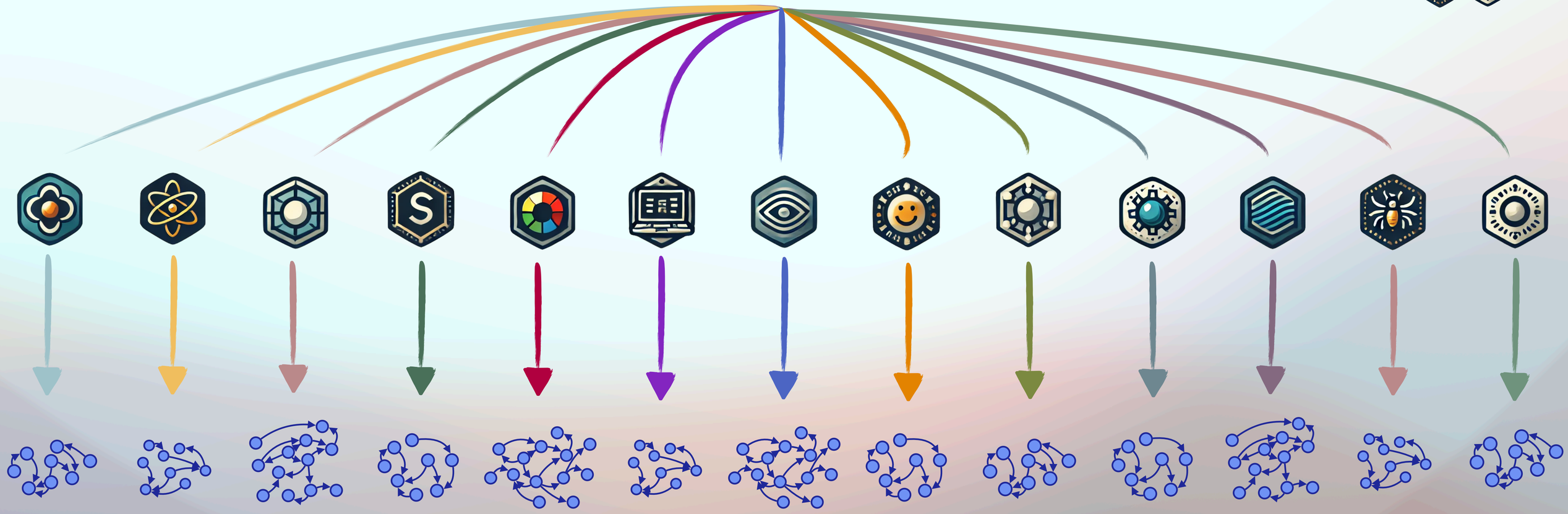
Static Analysis



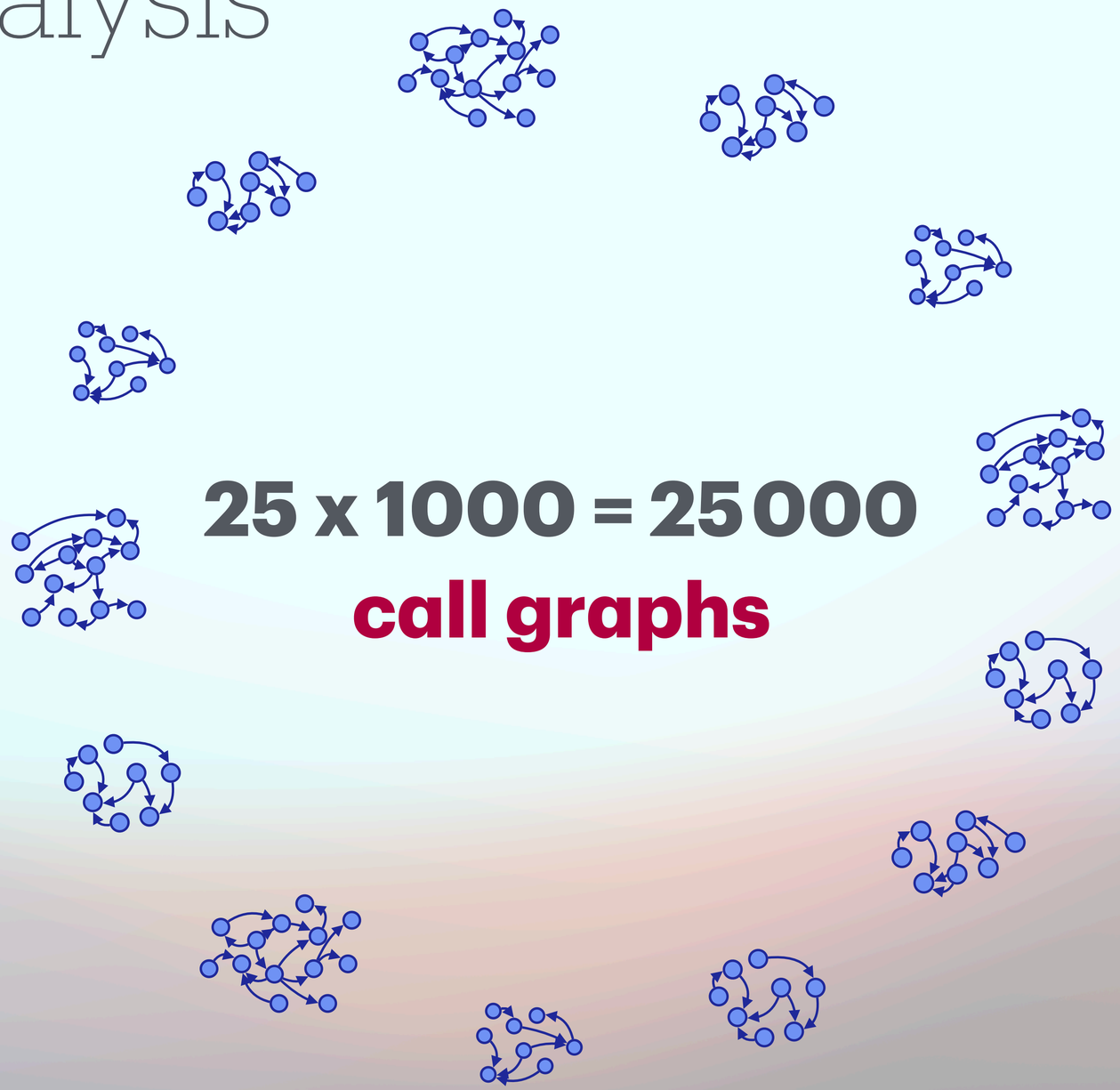
Static Analysis

When possible, we parametrized the call graph construction algorithm :

25 configurations



Static Analysis



25 x 1000 = 25 000

call graphs

126

Apps successfully analyzed by all tools

25 x 126 = 3150

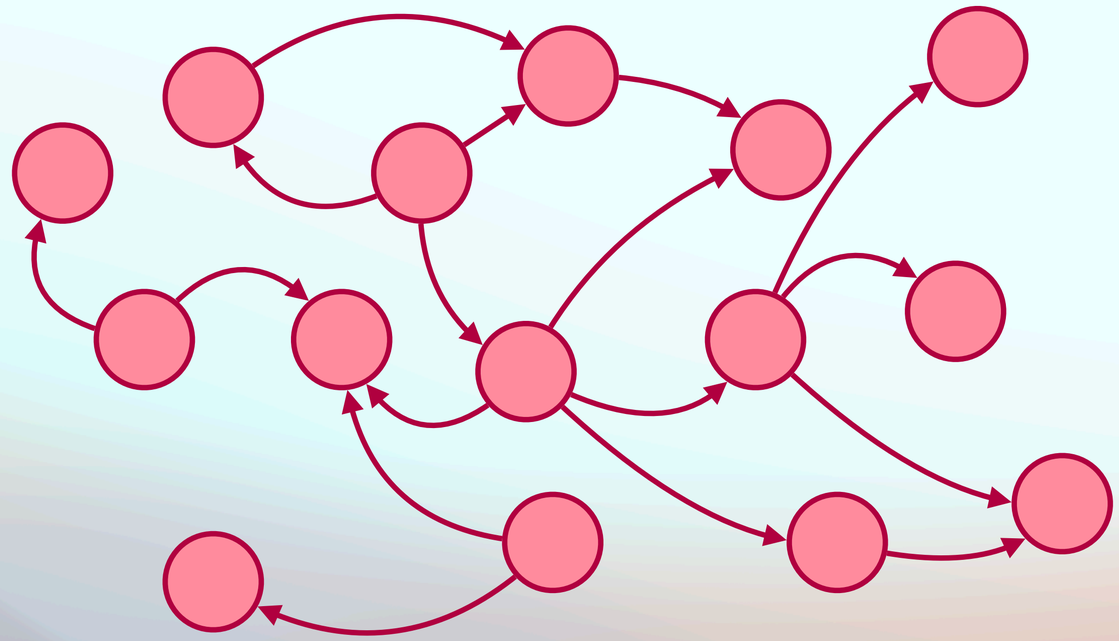
call graphs

		With libraries			Without libraries		
		Avg. $\|SM\|$	% M. in CG	Avg. $\|SE\|$	Avg. $\|SM^{\neg l}\|$	% M. in CG	Avg. $\|SE^{\neg l}\|$
FlowDroid	CHA	71 051	38%	399 975	6651	66%	48 218
	RTA	71 046	24%	227 493	6651	52%	33 802
	VTA	71 045	18%	109 519	6651	42%	16 788
	SPARK	71 031	5%	15 250	6649	12%	2391
IccTA	CHA	71 051	38%	399 981	6651	66%	48 220
	RTA	71 046	24%	227 541	6651	52%	33 746
	VTA	71 045	18%	109 023	6651	41%	16 703
	SPARK	71 031	5%	15 249	6649	12%	2391
RAICC	CHA	71 051	38%	397 791	6651	66%	47 894
	RTA	71 046	24%	224 574	6651	52%	33 271
	VTA	71 045	19%	111 151	6651	41%	16 605
	SPARK	71 031	6%	16 264	6650	12%	2434
DroidRA	CHA	71 053	38%	397 872	6652	66%	47 903
	RTA	71 048	24%	224 992	6652	52%	33 452
	VTA	71 047	19%	111 188	6652	42%	16 749
	SPARK	71 033	6%	16 437	6650	12%	2491
NatiDroid	CHA	61 758	81%	469 025	4837	88%	40 398
MaMaDroid	SPARK	60 500	5%	12 592	4791	14%	2007
BackDroid	SPARK	60 500	5%	12 592	4791	14%	2007
SootFX	SPARK	61 707	0%	101	4798	1%	9
ACID	SPARK	61 707	8%	54 169	4798	48%	4124
Gator	CHA	110 824	73%	1 920 412	31 342	90%	655 813
Jicer	SPARK	71 144	6%	15 763	6651	11%	2302
ArpDroid	SPARK	60 500	5%	12 593	4791	14%	2007
Difuzer	CHA	60 567	34%	245 987	4809	65%	31 060

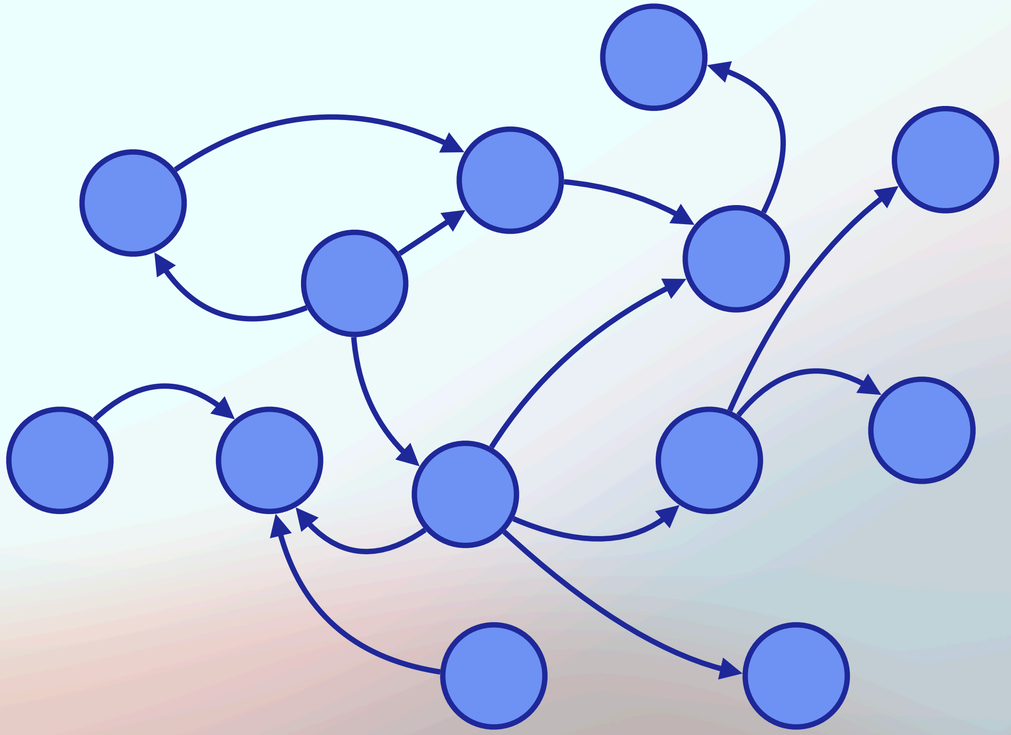
Comparison of static analysis tools

- Tools find different numbers of methods in apps
- Some tools supposed to add edges have **fewer edges** than baselines
- More precise call graph algorithms lead to significantly **fewer edges** in the call graph
- Tools consider a large proportion of apps as dead code
- The same call graph construction algorithm leads to **different call graphs**

Comparison

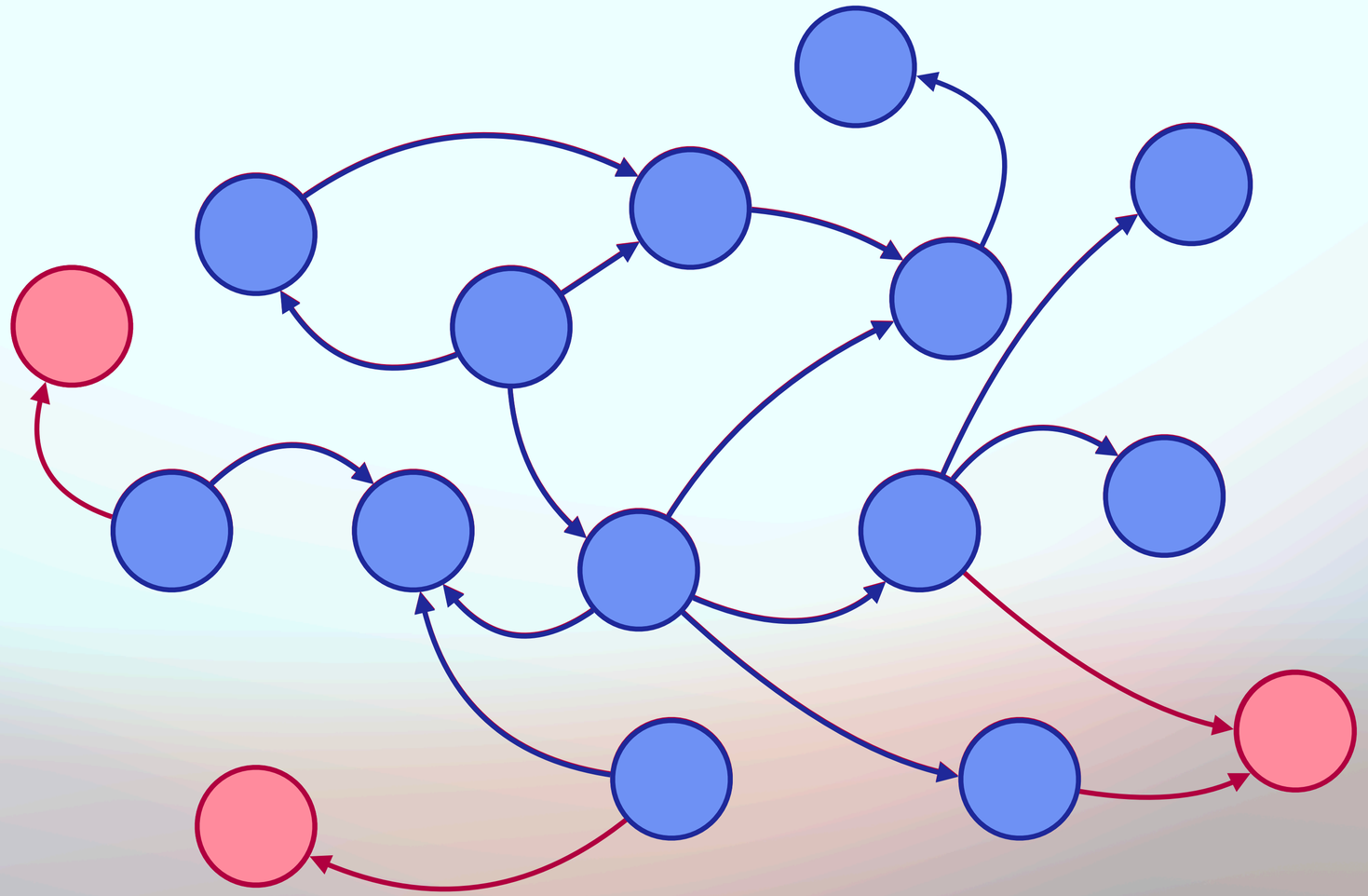


Dynamic Call Graph

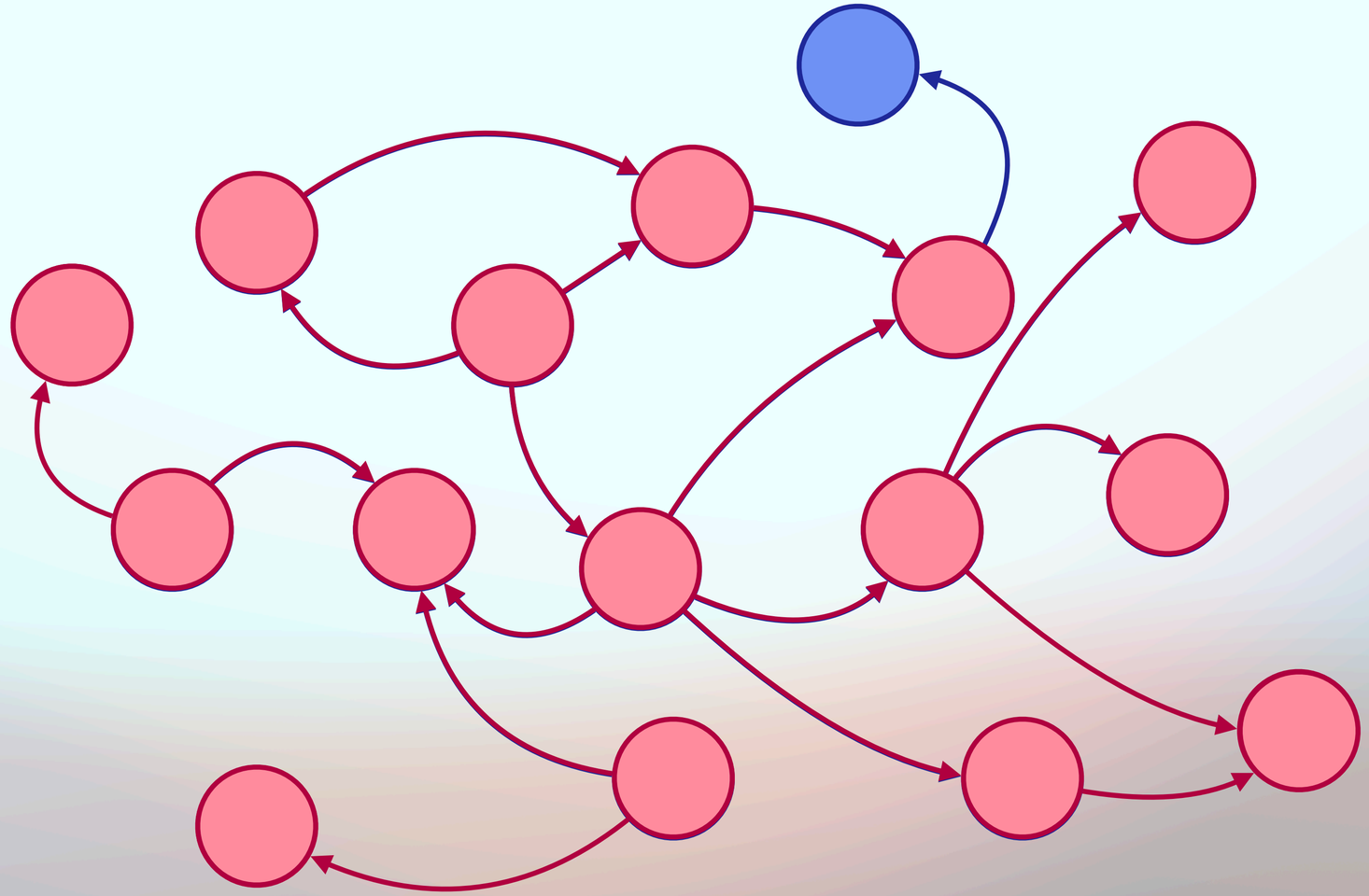


Static Call Graph

Comparison



Comparison



40%

methods missed with the
biggest over-approximation

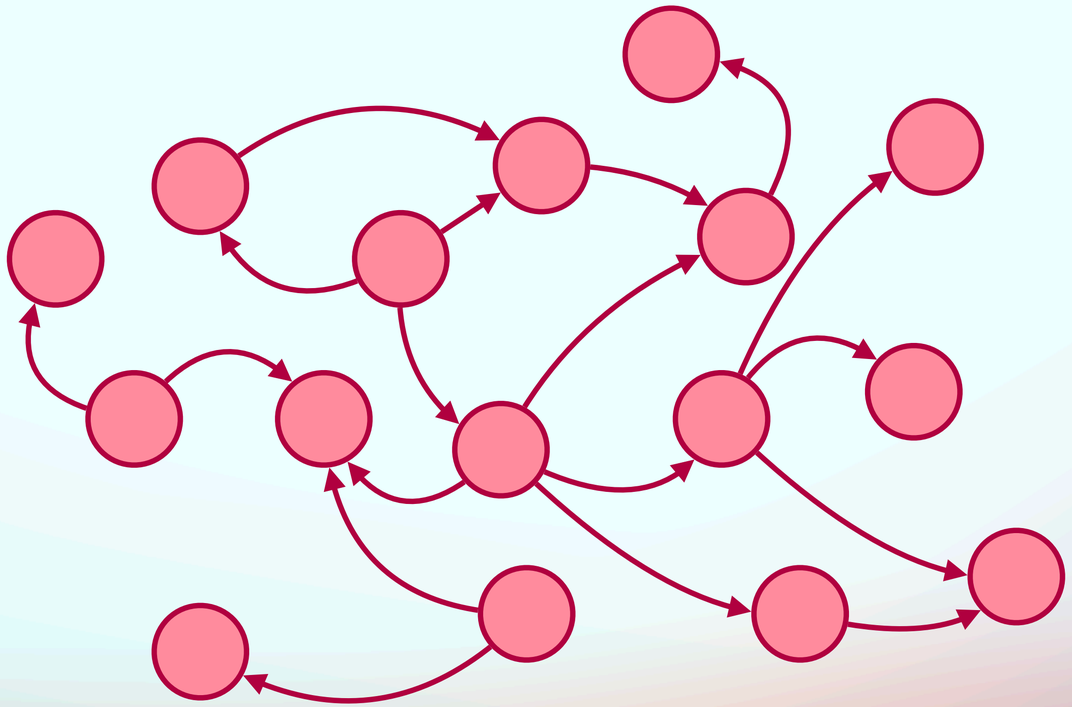
Comparison of dynamic and static analysis

- More precise call graph construction algorithms fail at their tasks
- The more precise an algorithm, the more **unsound**
- CHA-based tools have less **unsoundness**
- Even if CHA is the biggest over-approximation, it still falls short
- The bigger the code coverage does not mean the bigger the **unsoundness**

But!

What is the cause of this **unsoundness**?

Remember the dynamic call graph?



Remember the dynamic call graph?

They have no predecessor!

What do these nodes have in common?

We hypothesized that they are one of the main reasons for **unsoundness**

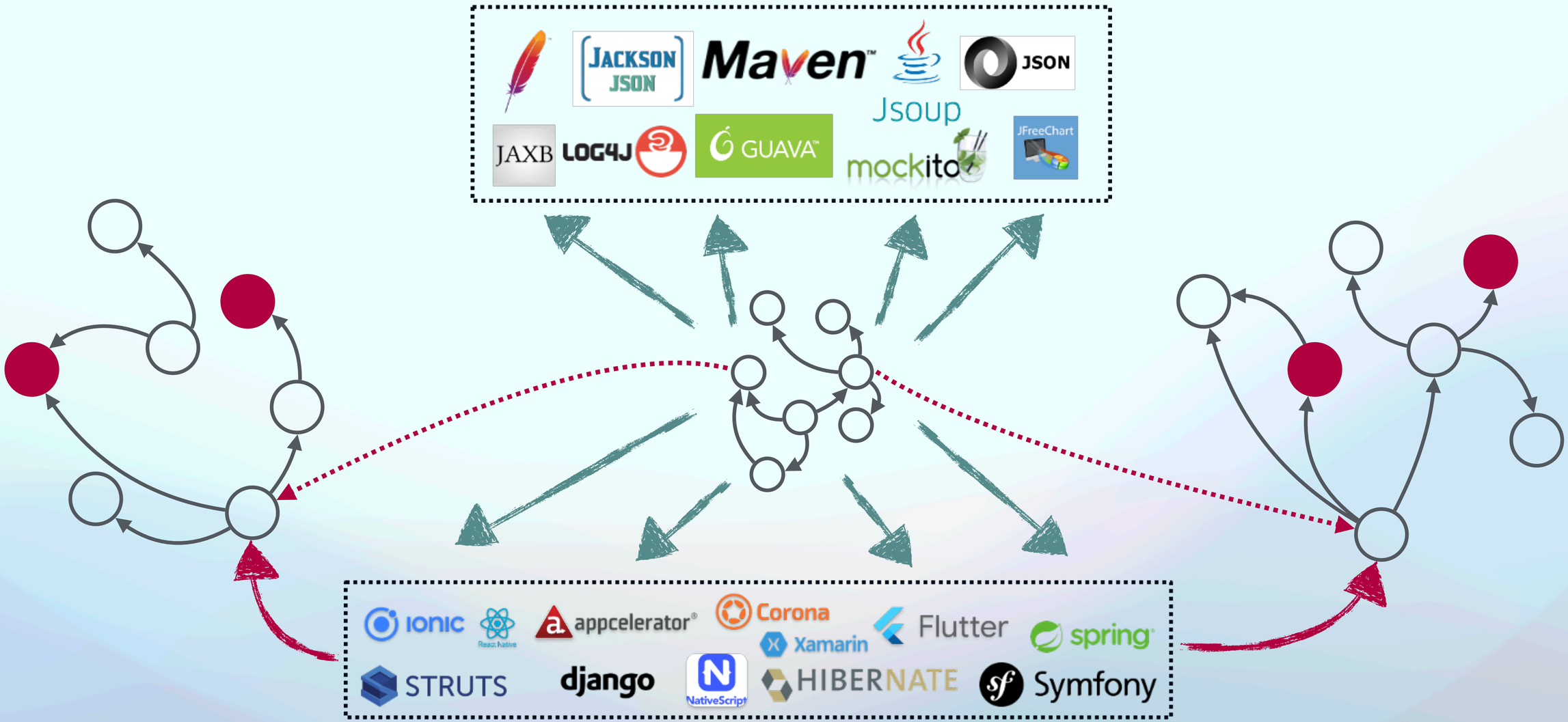
>16%

of methods do not have a predecessor, i.e., they are entrypoints

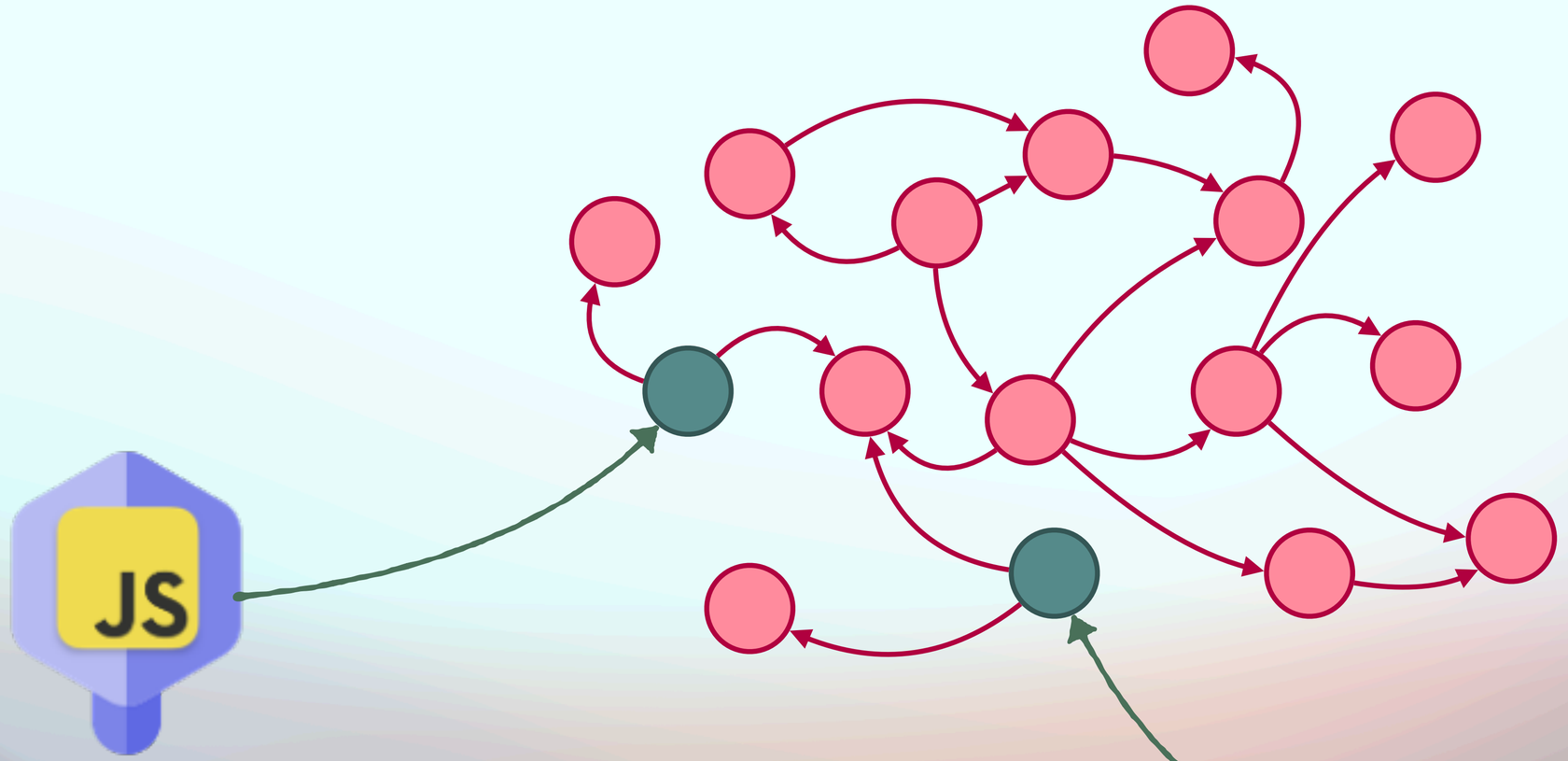
Causes of Unsoundness

- Many methods missed are derived from the **Android framework** methods
- Many methods missed are derived from **framework methods**, e.g., Google, Flutter, Ryanheise, or Unity3d
- All static analysis tools **miss at least 35% of these entry points**
- They represent **20% of all methods missed**
- Constructors, obfuscated methods, and lifecycle methods are among the most missed methods
- Methods indicative of **implicit mechanisms** are also among the most missed methods: onCreate, read, then, accept, onXXX, write***, apply, execute, etc.

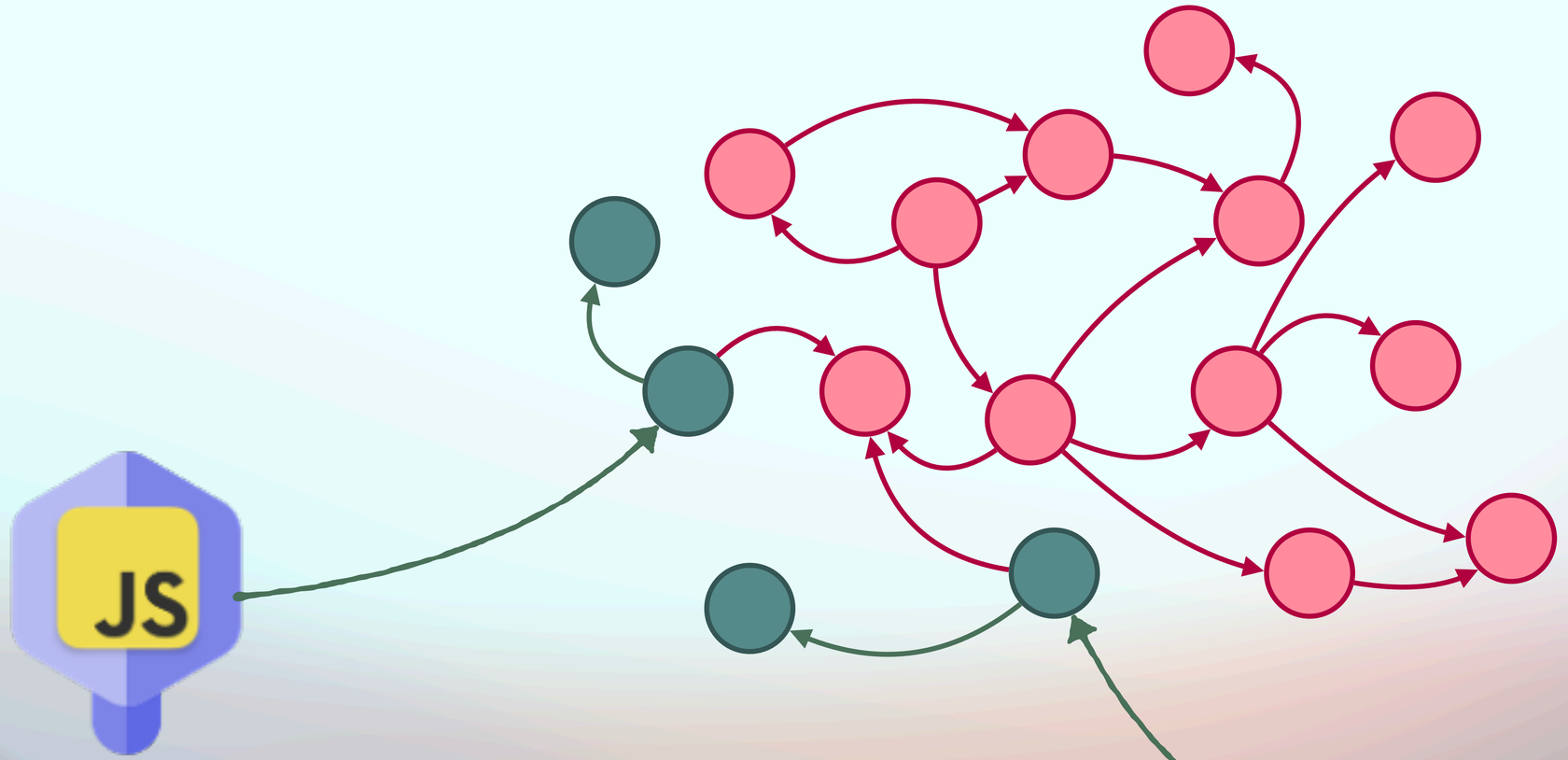
Frameworks


JACKSON JSON
Maven
JSON
Jsoup
JAXB
LOG4J
GUAVA
mockito
JFreeChart
IONIC
React Native
appcelerator
Corona
Xamarin
Flutter
spring
STRUTS
django
NativeScript
HIBERNATE
Symfony

Other languages



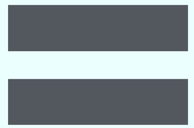
Flutter



Flutter

Implications for Security

Better Static Code Modeling

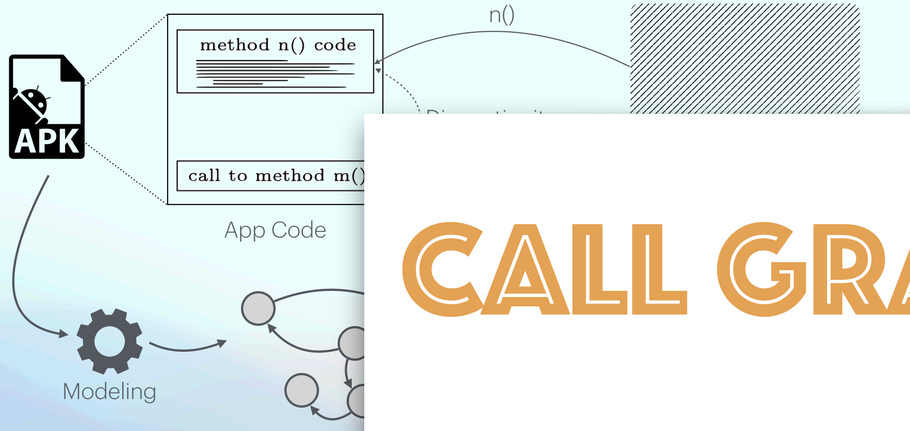


Better Static Code Coverage

Static analysis is NOT sound!

soundness of static analysis tools

Comprehensiveness of Program Analysis



September, 20th 2024 - Jordan Samhi

Dynamic Analysis

CALL GRAPH SOUNDNESS

https://github.com/JordanSamhi/Call-Graph-Soundness-in-Android-Static-Analysis

Static Analysis

...ts many
...research and
...proving the
...nalysis tools

Most of the icons used in this presentation come from: https://www.flaticon.com