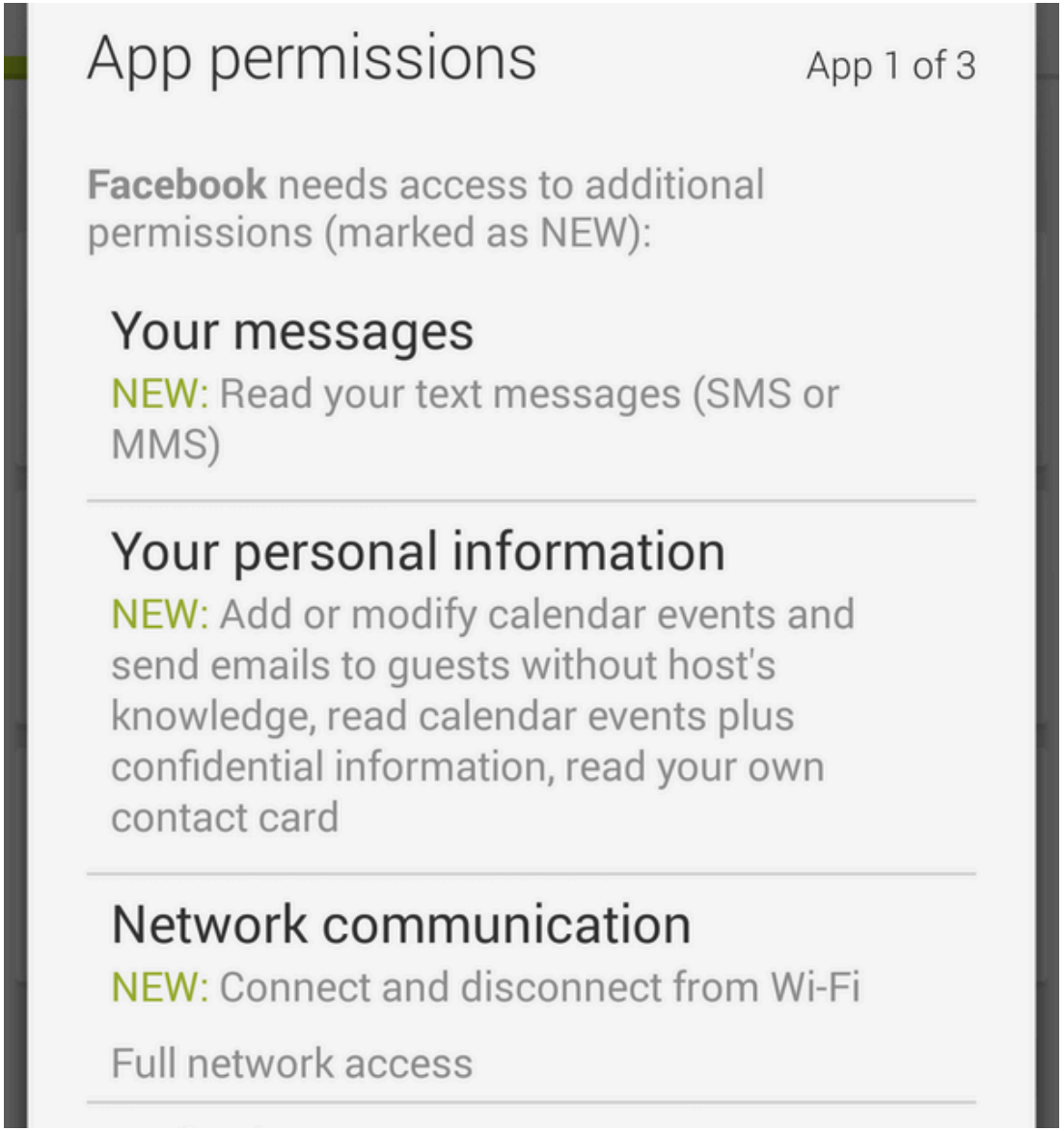
Effective Real-Time Android Application Auditing

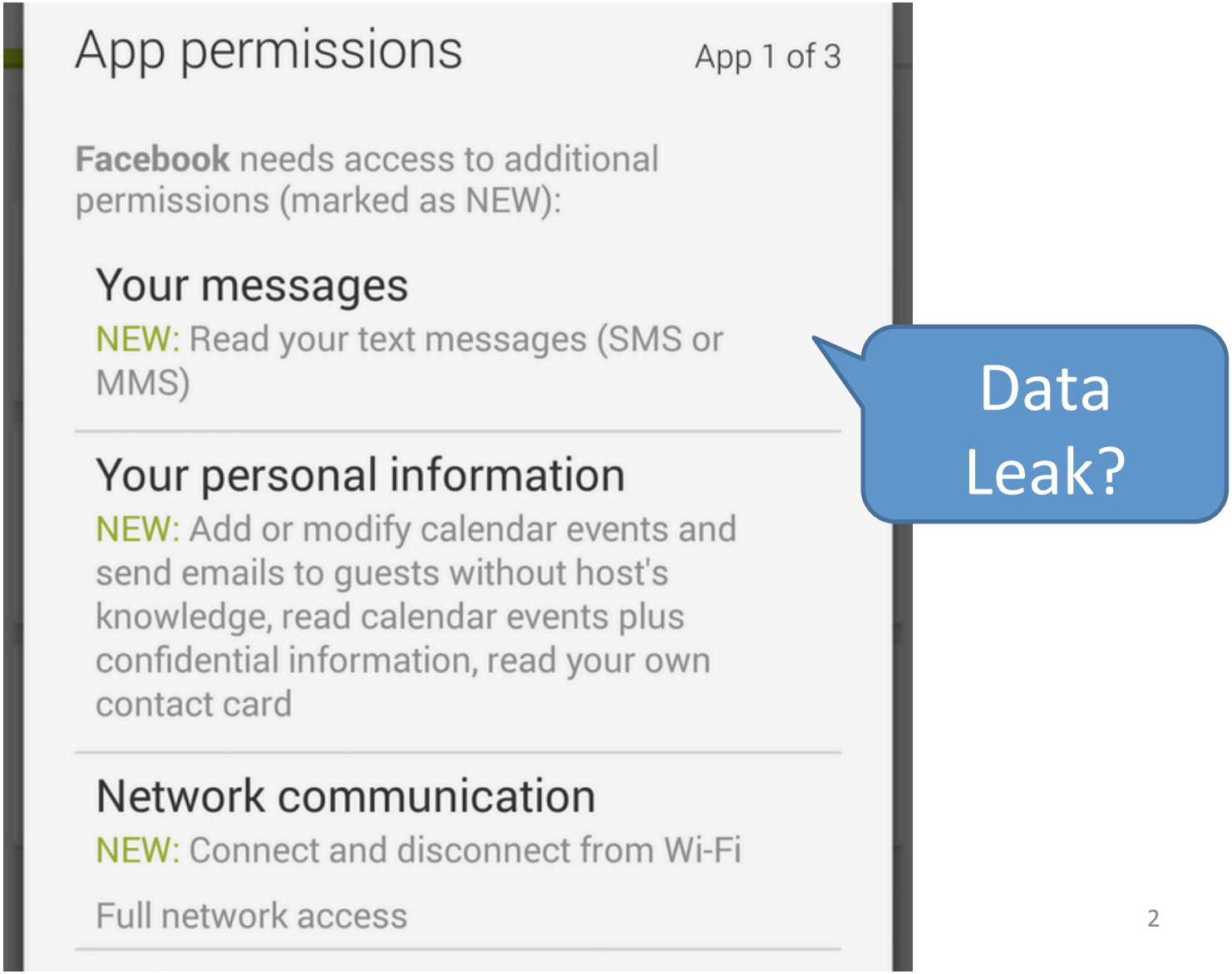
Mingyuan Xia, Lu Gong, Yuanhao Lyu,
Zhengwei Qi, Xue Liu

McGill University, Shanghai Jiao Tong University

App Permissions

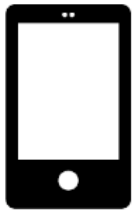


Data Leak?


App permissions
App 1 of 3
Facebook needs access to additional permissions (marked as NEW):
Your messages
NEW: Read your text messages (SMS or MMS)
Your personal information
NEW: Add or modify calendar events and send emails to guests without host's knowledge, read calendar events plus confidential information, read your own contact card
Network communication
NEW: Connect and disconnect from Wi-Fi
Full network access
Data Leak?

You Should!

Source and Sink



getPhoneNumber()
(source API)

“http://…&phone=5143980000”

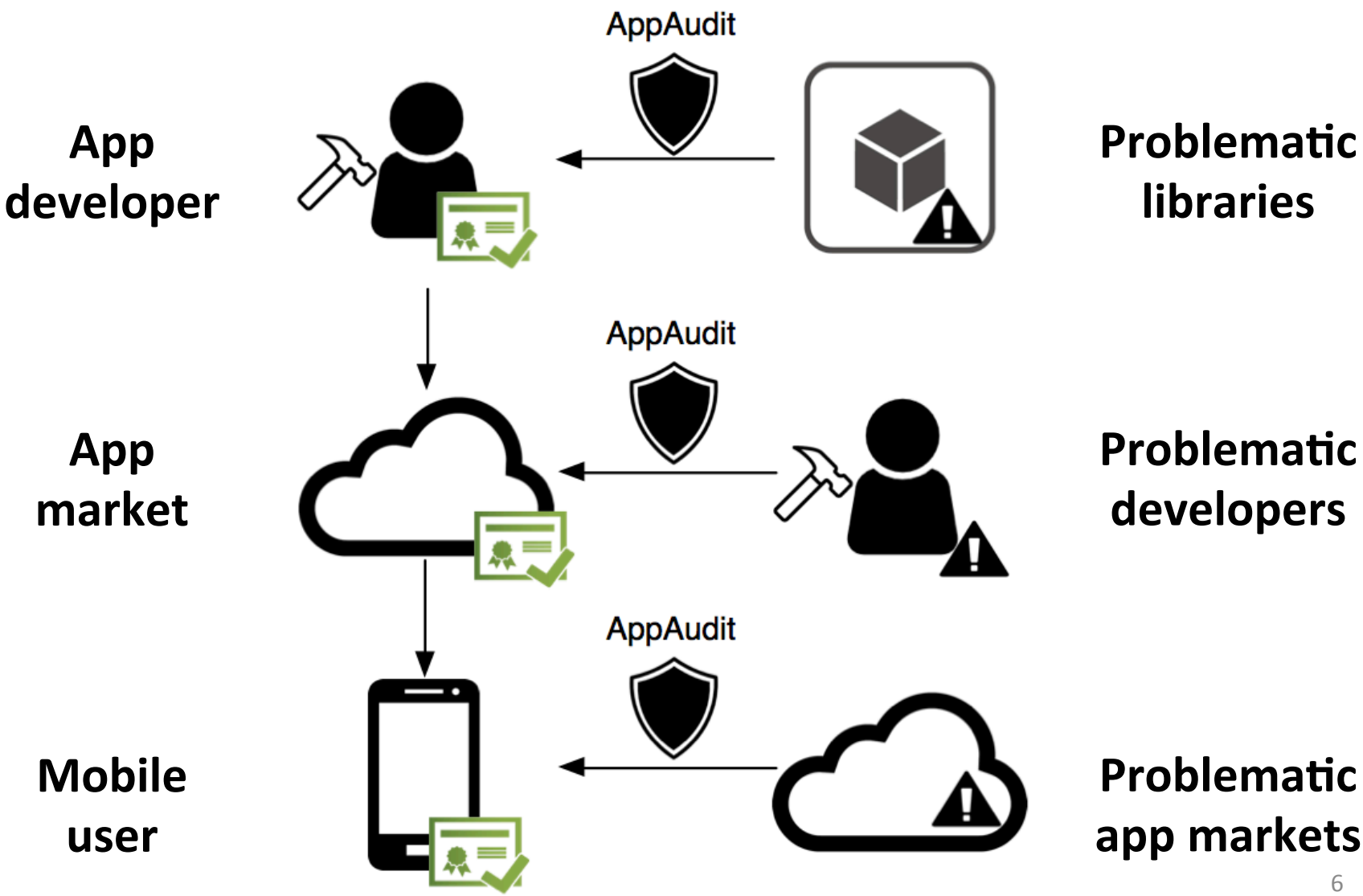
URL.openConnection(…)
(sink API)

App Auditing

Auditing Result

Item	Value
Source	Phone number
Sink	Network
Form	Unencrypted HTTP GET
Domain name	http:// …
Component	com.aa.bb

Use Cases


AppAudit
App developer
Problematic libraries
AppAudit
App market
Problematic developers
AppAudit
Mobile user
Problematic app markets

Existing Tools

App

- Dynamic analysis
  - e.g. TaintDroid [OSDI'10]

Existing Tools

App

- Dynamic analysis

  e.g. TaintDroid [OSDI'10]

  Limitations: dependent on user inputs and low coverage

Existing Tools

App

- Dynamic analysis

  e.g. TaintDroid [OSDI'10]

  Limitations: dependent on user inputs and low coverage

- Static analysis

  e.g. AppIntent[CCS'13]
  FlowDroid [PLDI'14]

Existing Tools

App

- Dynamic analysis

  e.g. TaintDroid [OSDI'10]

  Limitations: dependent on user inputs and low coverage

- Static analysis

  e.g. AppIntent[CCS'13]
  FlowDroid [PLDI'14]

  Limitations: false positives, time-consuming

Static Analysis Meets Real Apps

App

- Today's real apps
  - 10,000 ~ 100,000 functions
  - Millions of instructions

Static Analysis Meets Real Apps

App

- Today's real apps
  - 10,000 ~ 100,000 functions
  - Millions of instructions
- Today's static analysis (whole-program analysis)
  - Time: minutes ~ hours
  - Memory: 4GB ~ 32GB

AppAudit

- Problems:
  - Static analysis is slow
  - Dynamic analysis does not run automatically

AppAudit

- Problems:
  - Static analysis is slow
  - Dynamic analysis does not run automatically

- AppAudit design: A synergy of two analyses
  - Make static analysis faster with some false positives
  - Automates dynamic analysis, prune false positives

AppAudit

- Problems:
  - Static analysis is slow
  - Dynamic analysis does not run automatically

- AppAudit design: A synergy of two analyses
  - Make static analysis faster with some false positives
  - Automates dynamic analysis, prune false positives

Approximated execution

AppAudit

Dalvik bytecode

Coarse-grained static analysis

Suspicious functions

AppAudit

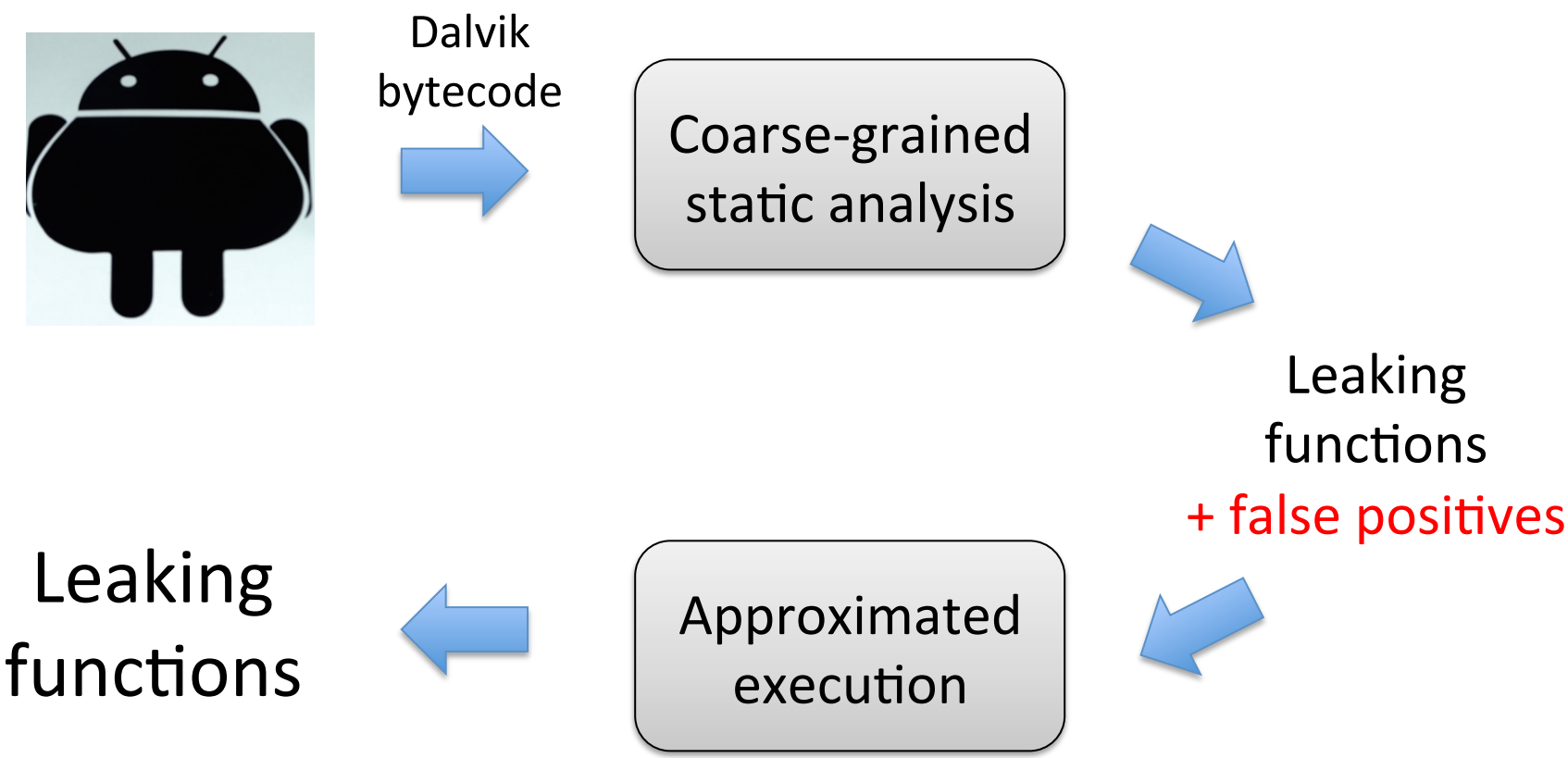
Dalvik bytecode

Coarse-grained static analysis

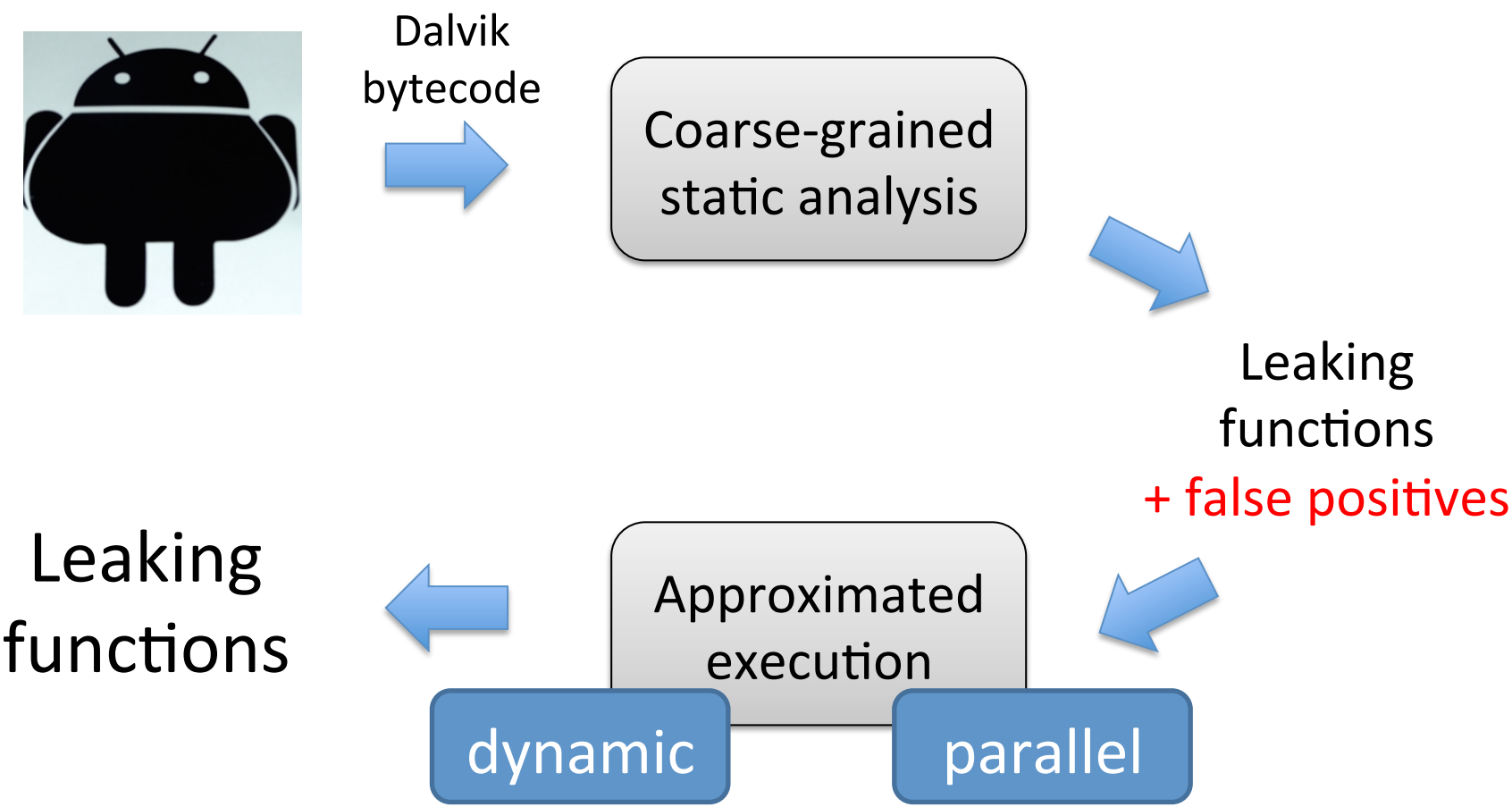
Leaking functions

+ false positives

AppAudit

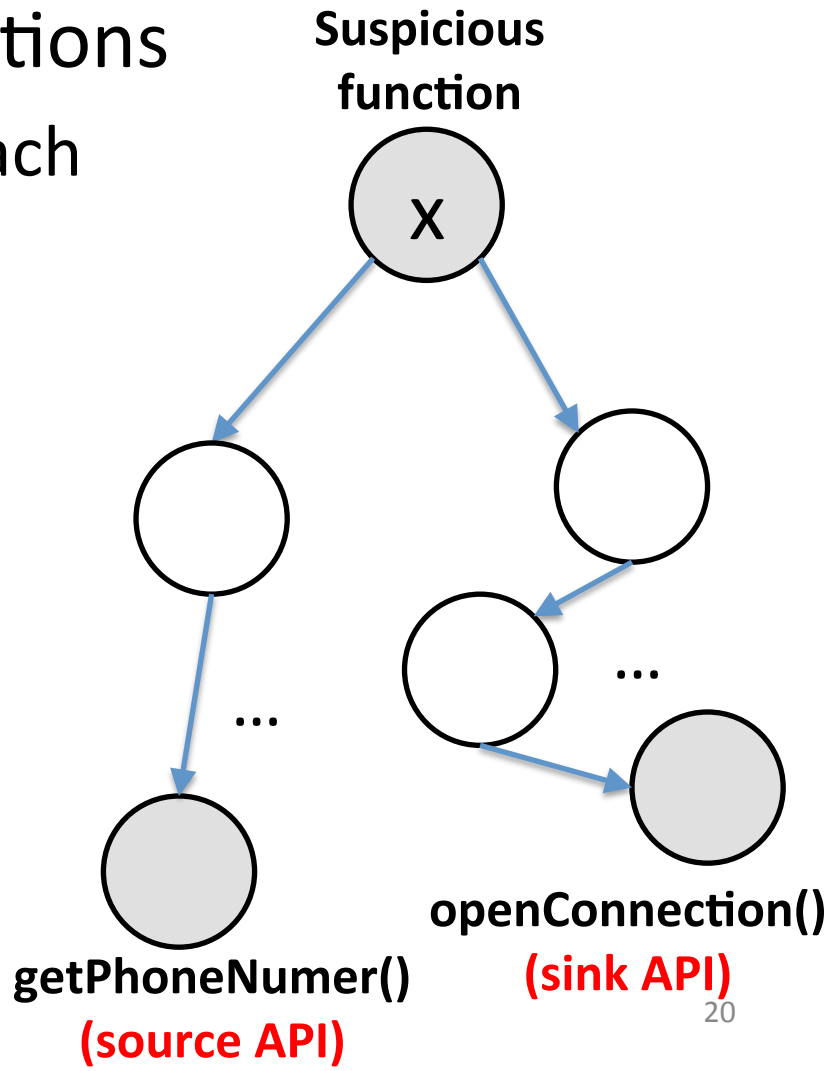

Dalvik bytecode
Coarse-grained static analysis
Leaking functions
+ false positives
Approximated execution
Leaking functions

AppAudit


Dalvik bytecode
Coarse-grained static analysis
Leaking functions
+ false positives
Approximated execution
dynamic
parallel
Leaking functions

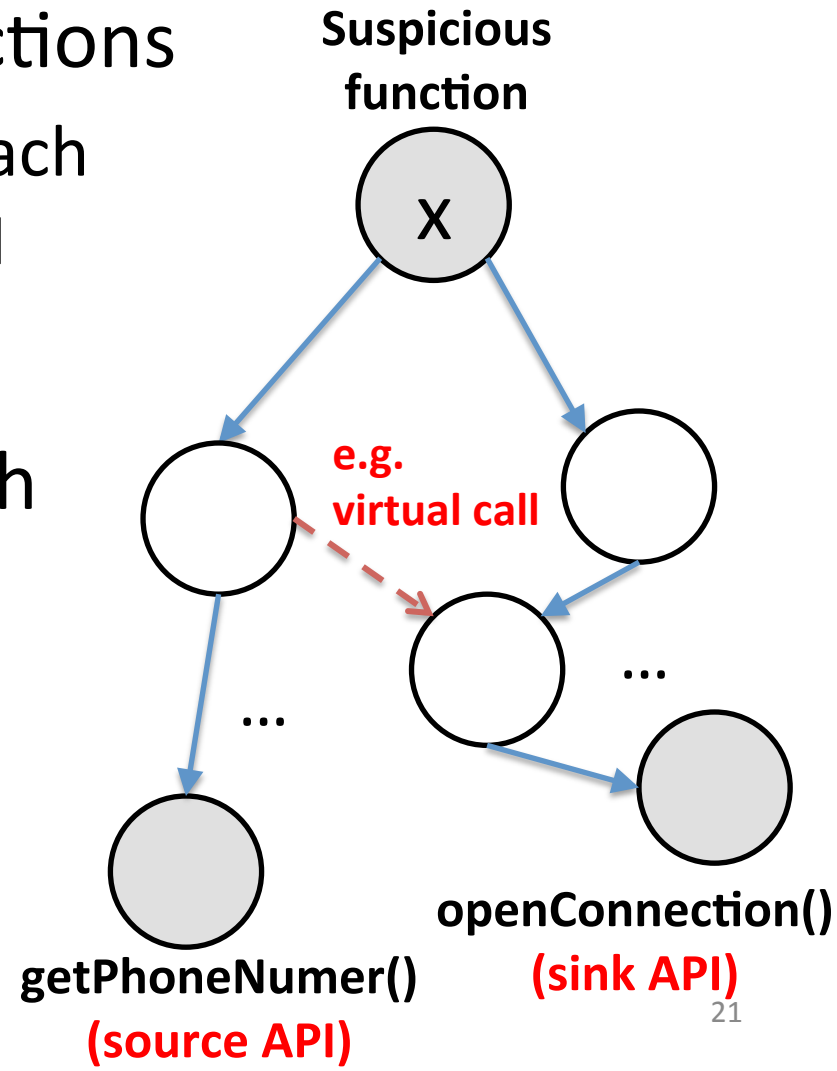
Lightweight Static Analysis

- Goal: find suspicious functions
  - Find functions that can reach both a source and sink API



Lightweight Static Analysis

- Goal: find suspicious functions
  - Find functions that can reach both a source and sink API
- Over-estimating call graph
  - Efficient
  - Some false positives



Dynamic Analysis

- Approximated execution
  - Automatically and approximately execute a suspicious function to confirm leaks
  - Mimic real execution

Dynamic Analysis

- Approximated execution
  - Automatically and approximately execute a suspicious function to confirm leaks
  - Mimic real execution

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
```

unknown

A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```

	type	value
x:	?	?

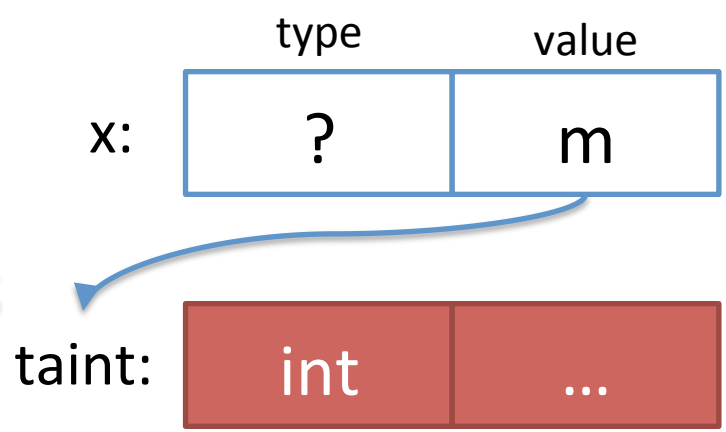
A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();   // getPhoneNumber()
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```

	type	value
x:	?	?
taint:	int	...

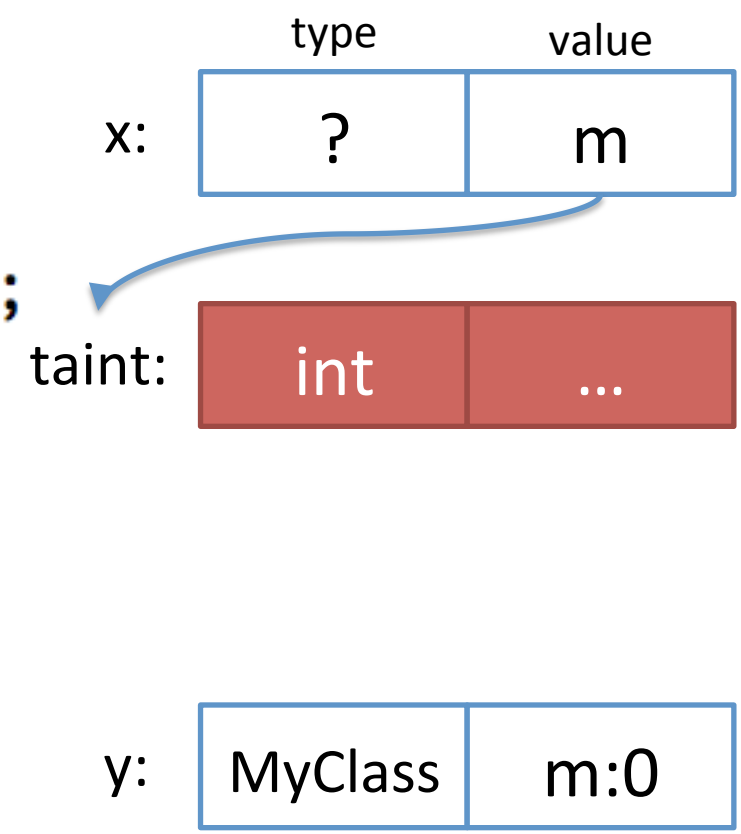
A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```



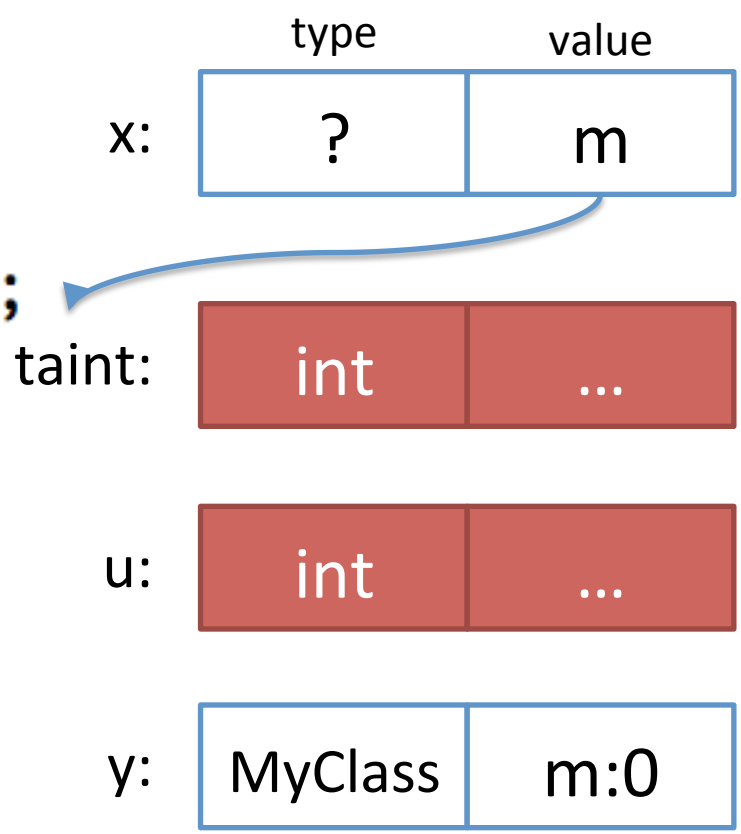
A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```



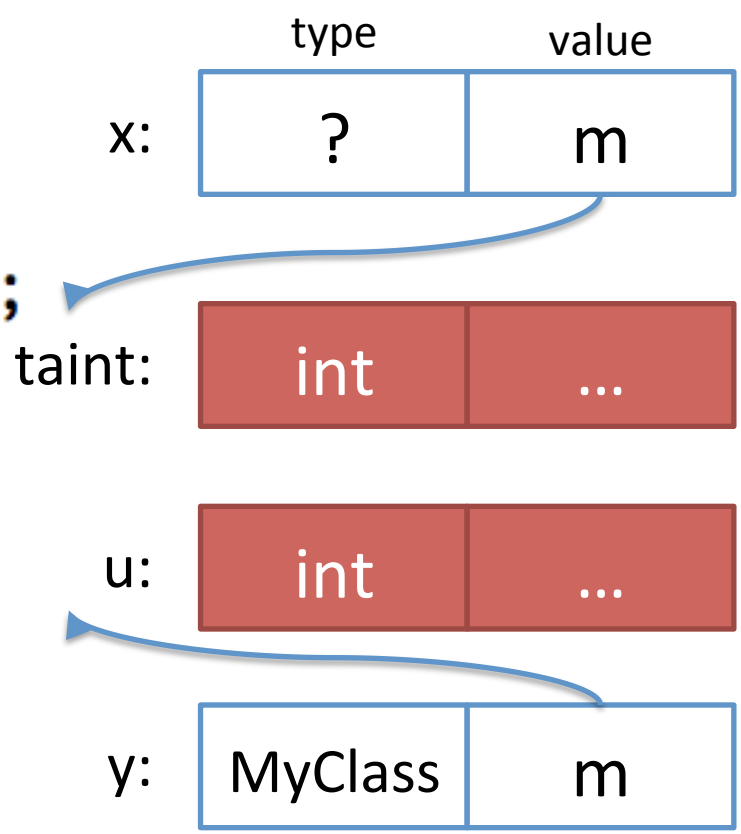
A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```



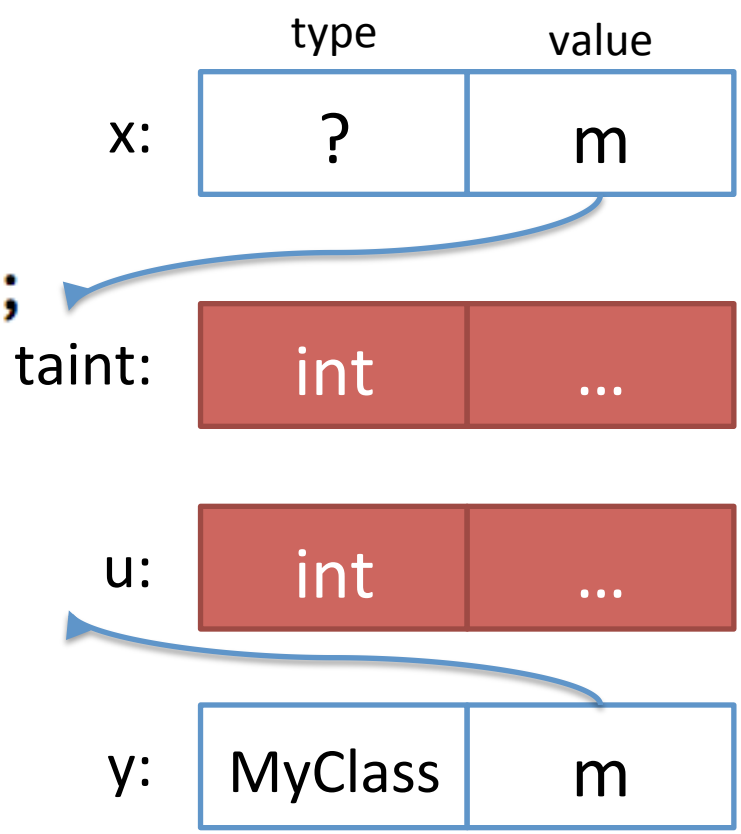
A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```



A Running Example

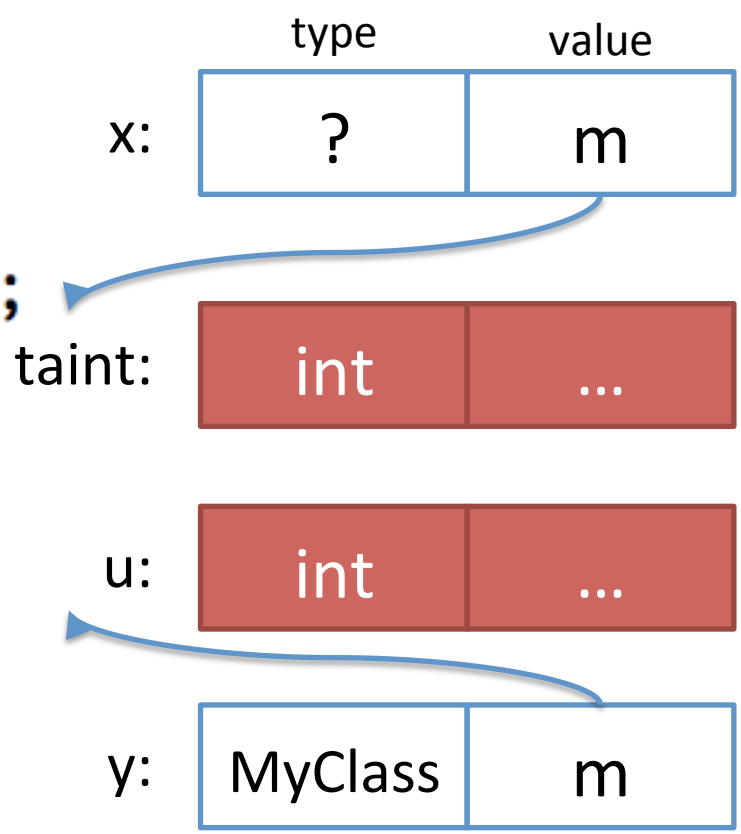
```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```



A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);
}
```

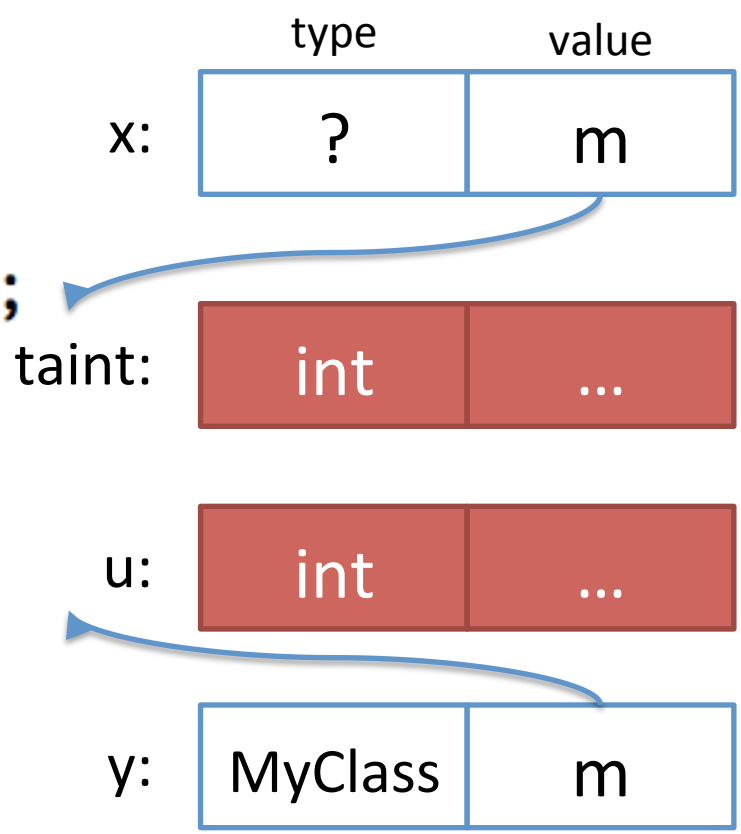
True



A Running Example

```
public static class MyClass {
    int m;
}
static void foo(MyClass x) {
    x.m = source();
    MyClass y = new MyClass();
    int u = ~x.m;
    y.m = u;
    if (x != y)
        sink(y.m);  // send()
}
```

Sink() meets a taint, leak!



Approximated Execution

- Executing bytecode
  - 15 rules for manipulating unknown objects

Approximated Execution

- Executing bytecode
  - 15 rules for manipulating unknown objects
  - Keep executing until
    - finding a leak
    - function terminated
    - encountering an unknown control flow -> in the paper

Approximated Execution

- Executing bytecode
  - 15 rules for manipulating unknown objects
  - Keep executing until
    - finding a leak
    - function terminated
    - encountering an unknown control flow -> in the paper

✓ **No false positives**

✓ **Deterministic leaks**

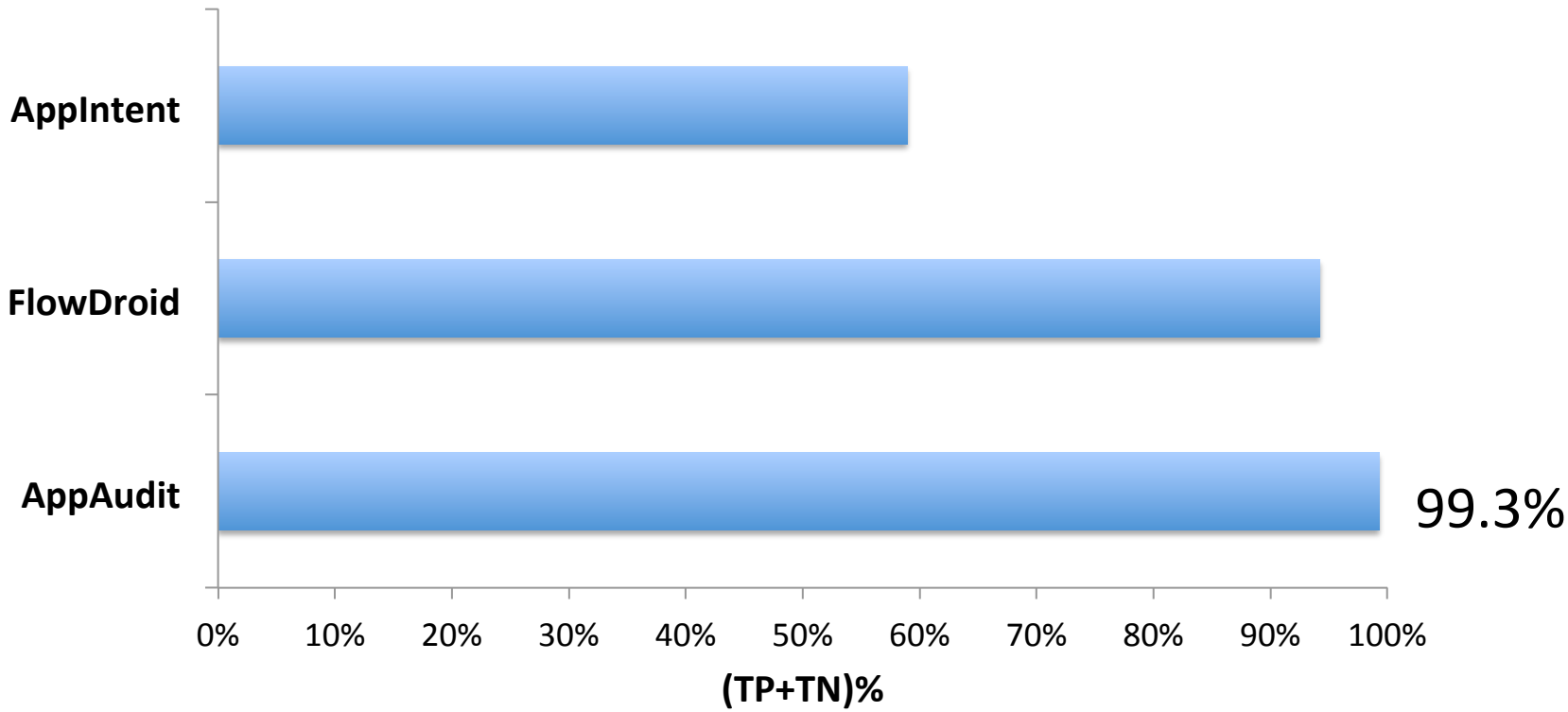
? **External input dependent leaks**

Evaluation

- Correctness & Accuracy
  - Malware genome datasets [S&P'12]
  - Droidbench: well-crafted cases against analysis
- Analysis time & Memory cost
  - Malware and real apps
- Some real leaks found in real apps

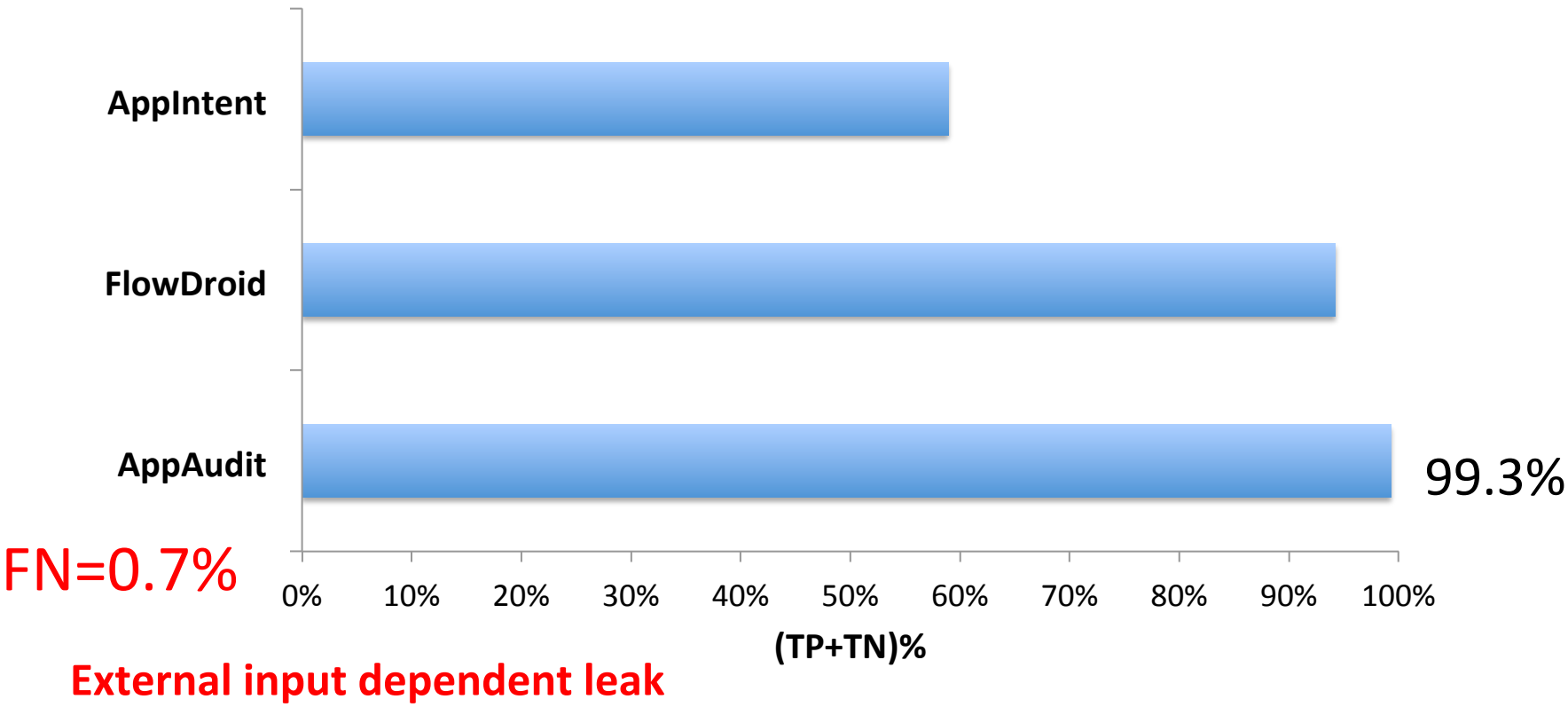
Accuracy

- Malware dataset: 1,004 positives + 1 negative



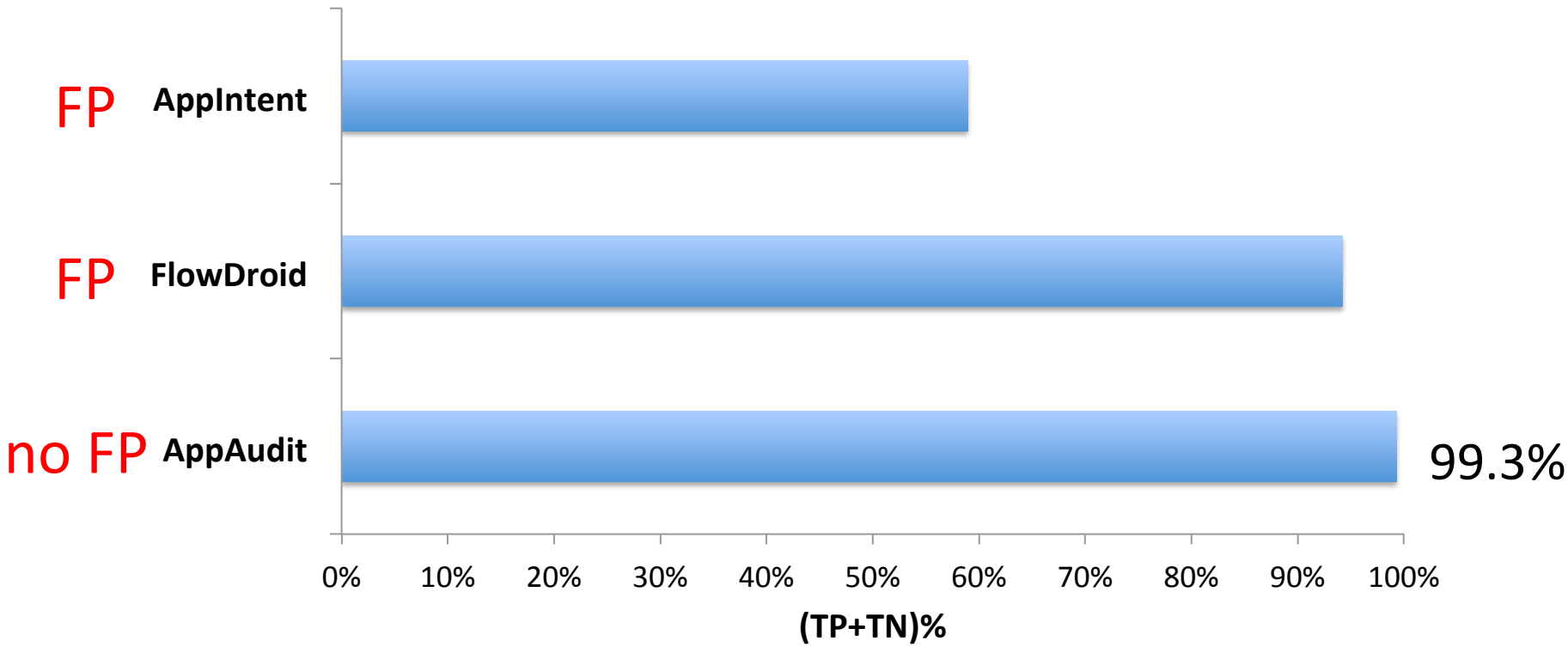
Accuracy

- Malware dataset: 1,004 positives + 1 negative

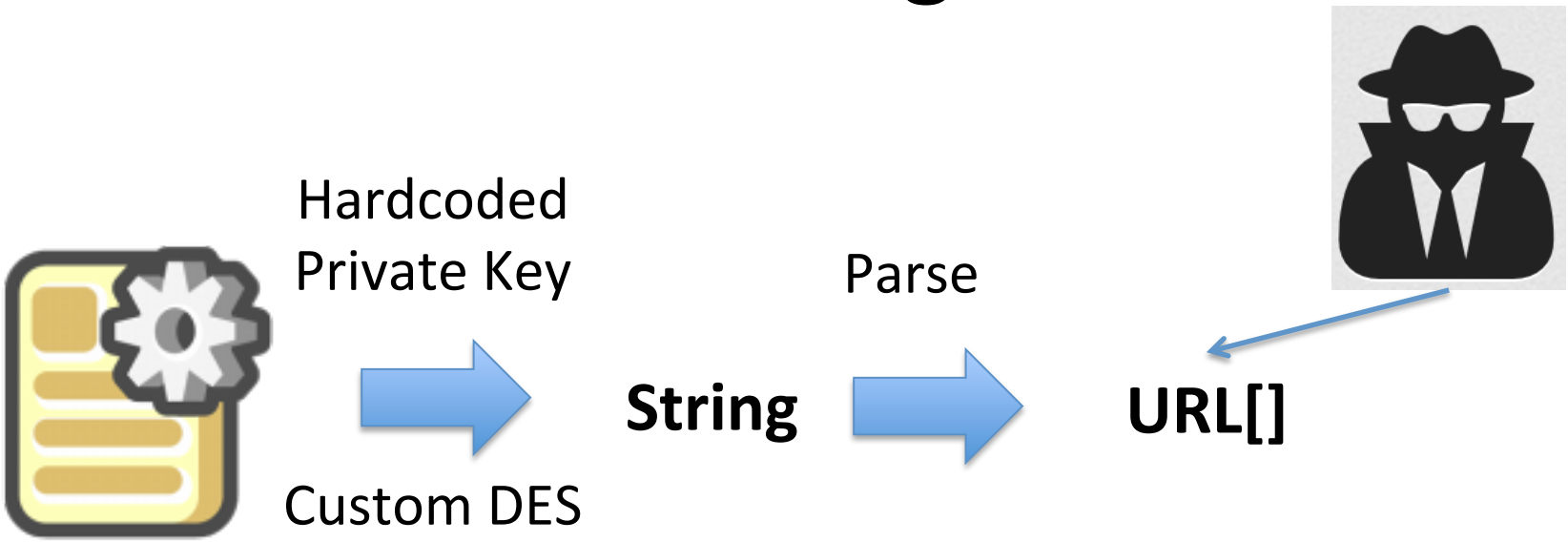


Accuracy

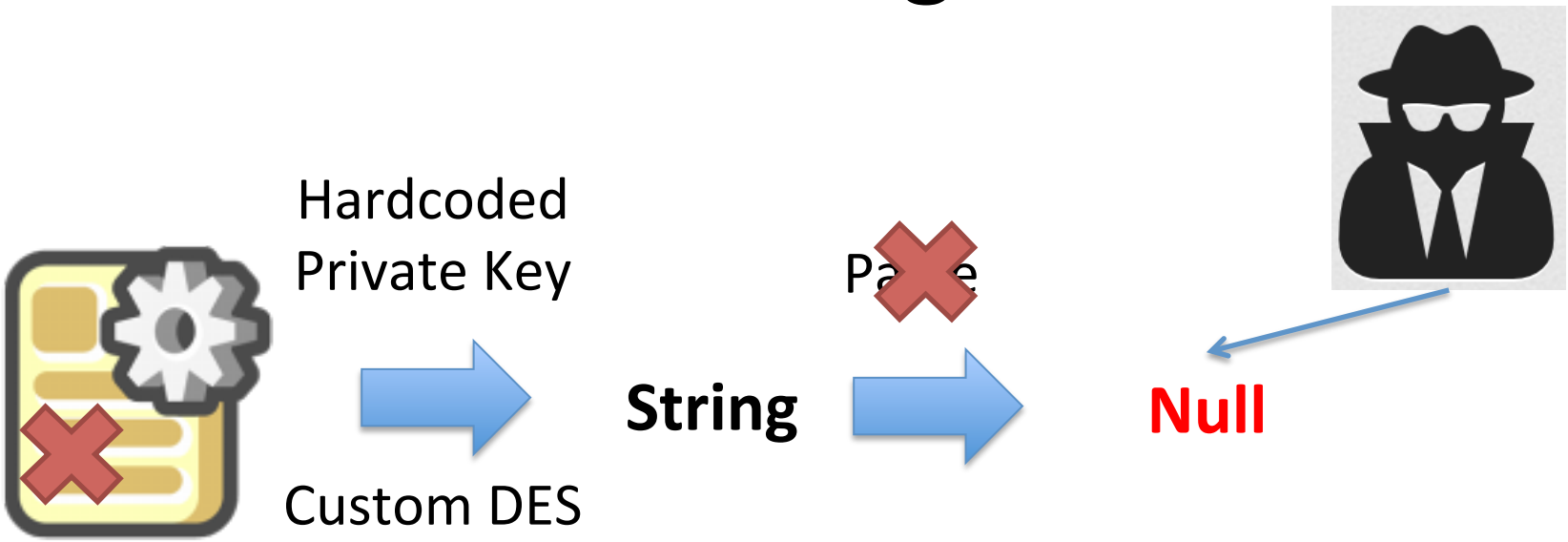
- Malware dataset: 1,004 positives **+ 1 negative**



The True Negative

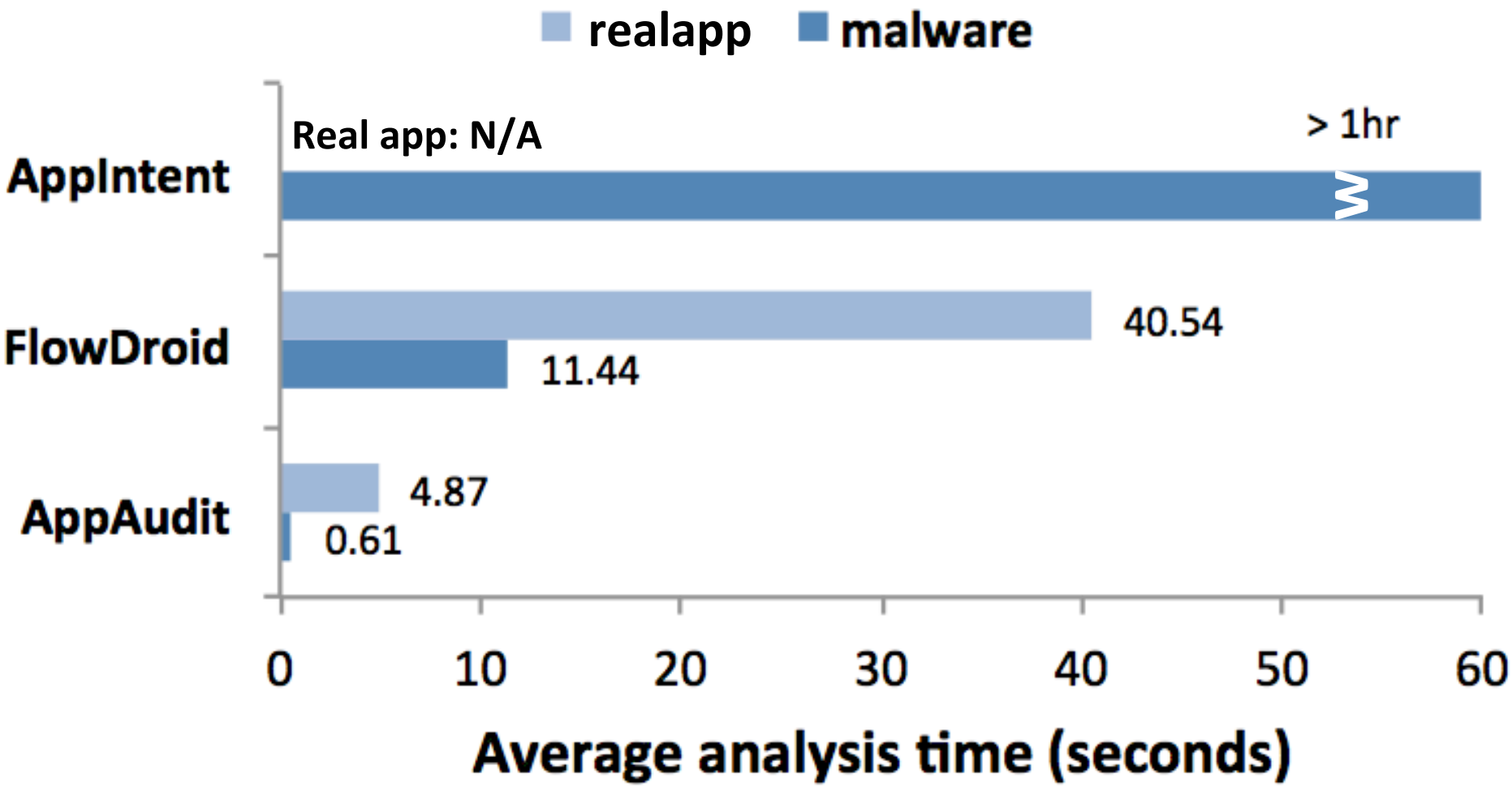


The True Negative

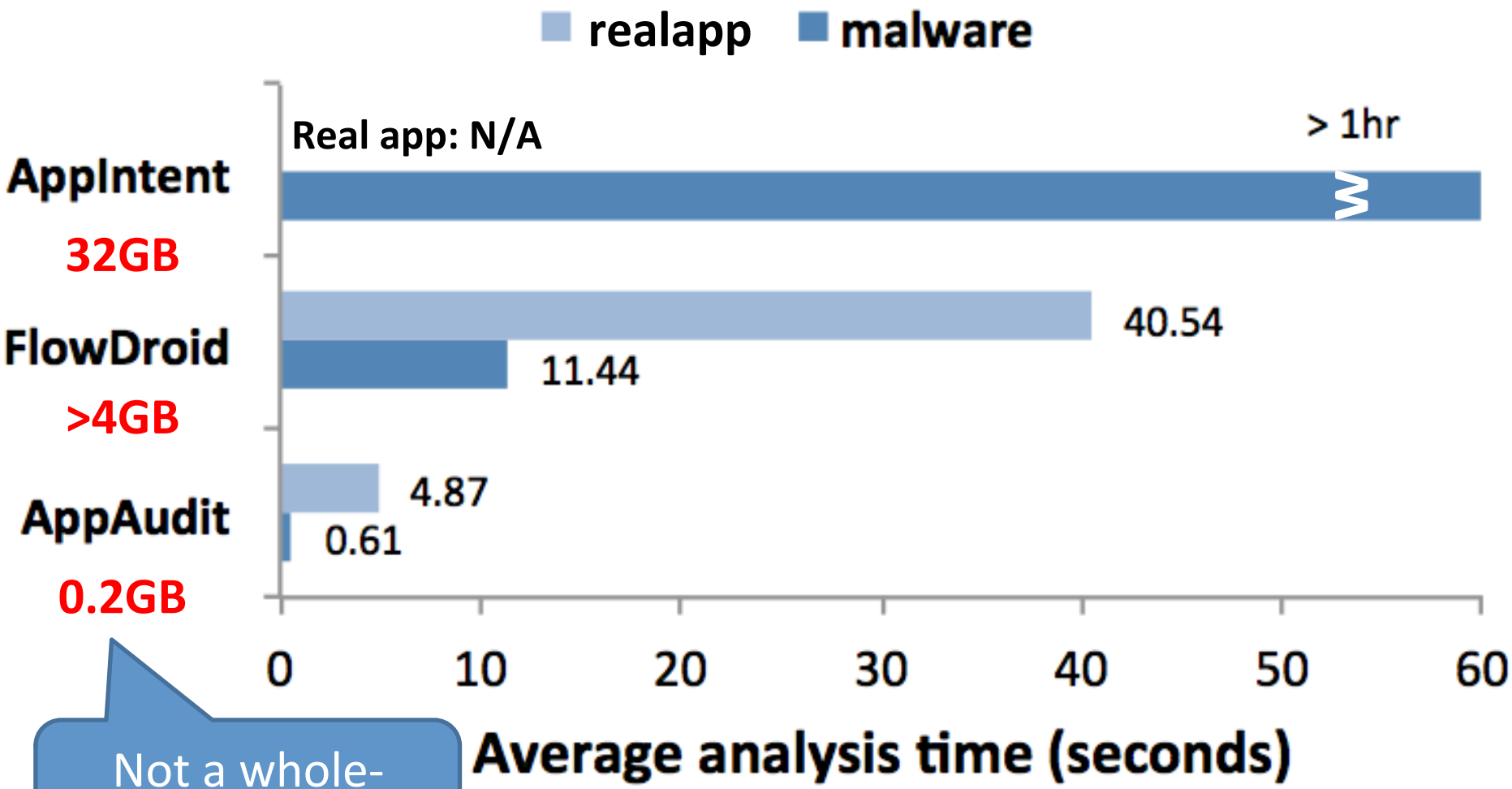


Malformed configuration file disables the malware

Analysis Time

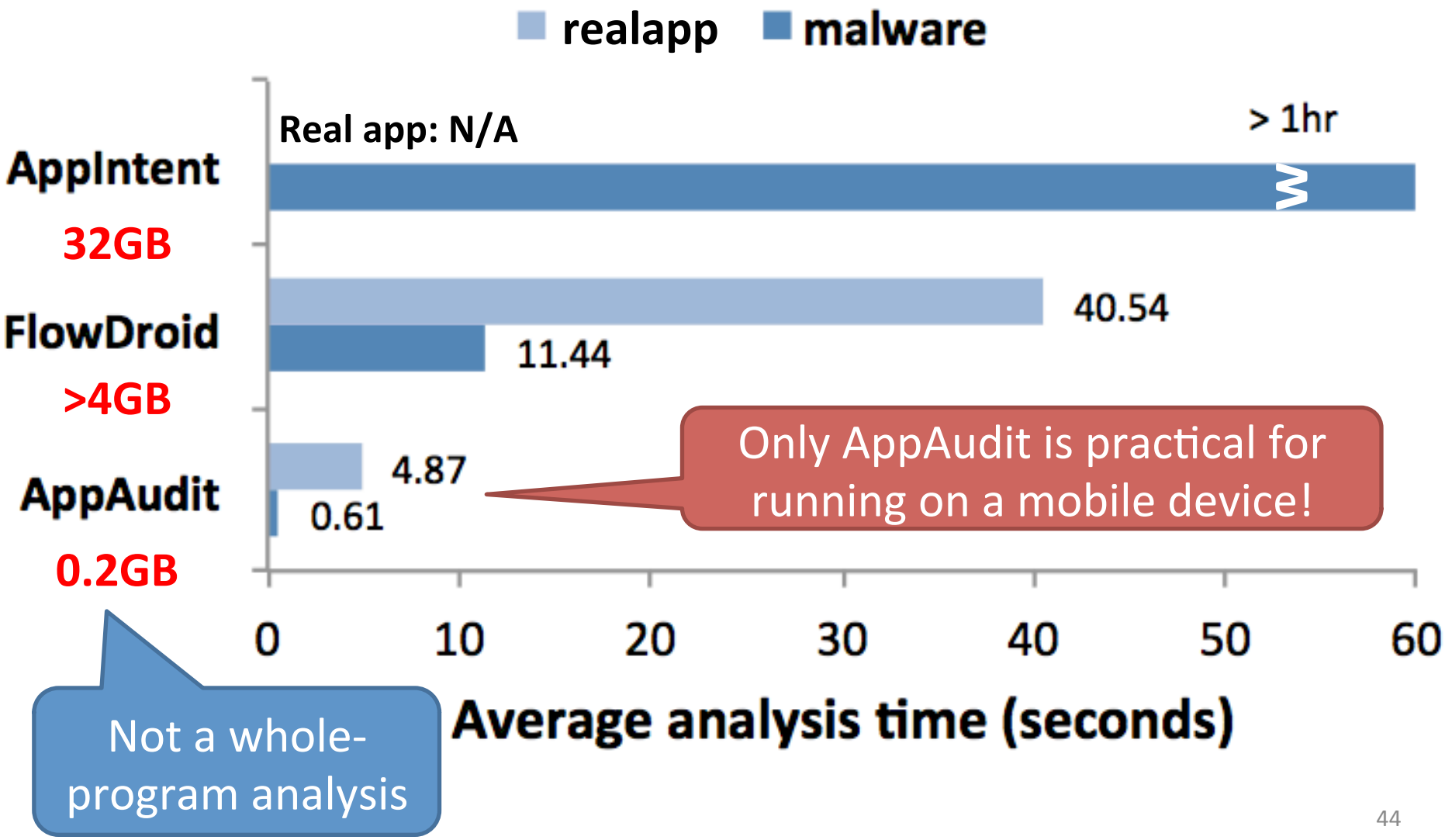


Memory Cost



Not a whole-program analysis

Memory Cost


realapp
malware
Real app: N/A
> 1hr
AppIntent
32GB
FlowDroid
>4GB
AppAudit
0.2GB
40.54
11.44
4.87
0.61
Only AppAudit is practical for running on a mobile device!
0
10
20
30
40
50
60
Average analysis time (seconds)
Not a whole-program analysis

400 real apps, 30 data leaks

No false positives
Manual confirmed

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	app,~~lib~~	50M-100M

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	app,~~lib~~	50M-100M

Total: half billion installs!

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	~~app,lib~~	50M-100M

What component leaks most?
Ad libraries

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	~~app,lib~~	50M-100M

Privacy Policy?
Only a few mention 3rd party libs

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	~~app,lib~~	50M-100M

What data are leaked most?
IMEI, IMSI, Location

Among ~15 taints we tracked

Real Leaks

Name	Component	Source	Venue	Privacy Policy	Installs (M for millions)
Texas Poker v4.0.1	App	Location	HTTP GET	×	10M-50M
Word Search v1.14	Mobfox	Location, IMEI	HTTP GET	~~app,lib~~	0.5M-1M
Speedtest v2.09	Mobfox	Location, IMEI	HTTP GET	app,lib	10M-50M
Brightest Flashlight v2.3.3	MDotm, Mobclix	IMEI, IMSI	HTTP GET	~~app,lib~~	50M-100M

How are data leaked?
HTTP (not encrypted!) and HTTPs

In the paper

400 real apps, 30 data leaks

http://appaudit.io

~~400~~ real apps, ~~30~~ data leaks

~140,000 **8,537**

http://appaudit.io

~~400~~ real apps, ~~30~~ data leaks

~140,000 **8,537**

2 days

Work-in-progress

- More auditing details
  - E.g. when the leak would happen
- Mobile version
- Developer tool
  - Android Studio
  - Embed for Github repo

Thank you

http://appaudit.io

AppAudit

Developers Research Contact

data leaker

富甲三國：百萬骰子王

air.com.A6fgame.newfjsgFTOLipad.android

version: 3.0.1

GooglePlay Rating:

3.6

5 ★ 1 ★

Leak phone identity

via a public file on the SDCARD

data leaker

AMC Security - Antivirus Clean

com.iobit.mobilecare

version: 4.5.0

GooglePlay Rating:

4.5

5 ★ 1 ★

Read SMS messages

Automatically start

Leak phone identity

You data will be encoded in an HTTP GET REQUEST and transmit to a remote server

data leaker

WangXin - Ali Mobile Taobao

com.alibaba.mobileim

version: 2.8.9

GooglePlay Rating:

3.7

5 ★ 1 ★

Automatically start

Leak phone identity

via an UNENCRYPTED HTTP GET REQUEST

Detect virus content

AegisLab detects MTK_1

Common Q/A

1. You have FNs. So it is not complete?
2. The definition of leaks?
3. iOS?
4. Native code?
5. What about paid apps?
6. Can AppAudit actually run on a phone?
7. What can I do if I know an app is leaking something?
8. Reflection, control-flow taints, taint sanitization, infinity, multi-threading, compiler sensitivity, file taints, how it actually works?

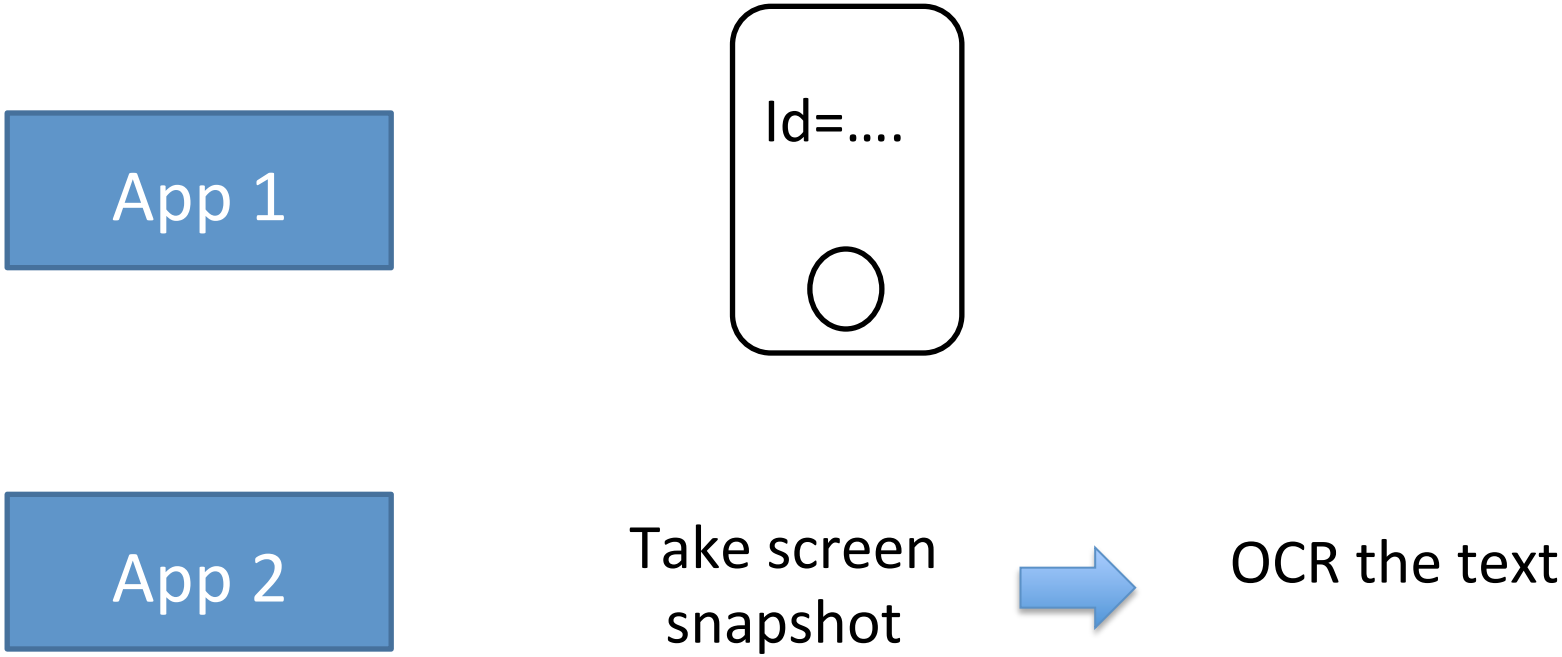
In the paper

Completeness

no false positive → Generative model AppAudit execution path
-> User inputs
-> Real execution path that leaks (TaintDroid?)

no false negative → Largely depend on the completeness model itself

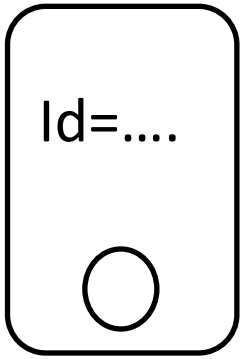
Completeness (cont.)



Inter-process, computer vision based leak

Completeness (cont.)

App 1



App 3

Front camera:
take picture

Human-phone, computer vision based leak