Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications

Amiangshu Bosu, Fang Liu, **Danfeng (Daphne) Yao,**
& Gang Wang

SECURITY

CIA to open private 'app store' for intelligence operatives via Amazon Web Services

Martin Anderson Fri 27 Feb 2015 2.18pm

G+ Y

Next month the US Central Intelligence A
provisioning from Amazon Web Services
Information Officer Doug Wolfe at the 4th
Wednesday.

U.S. Department of Defense to Open Its Own App Store

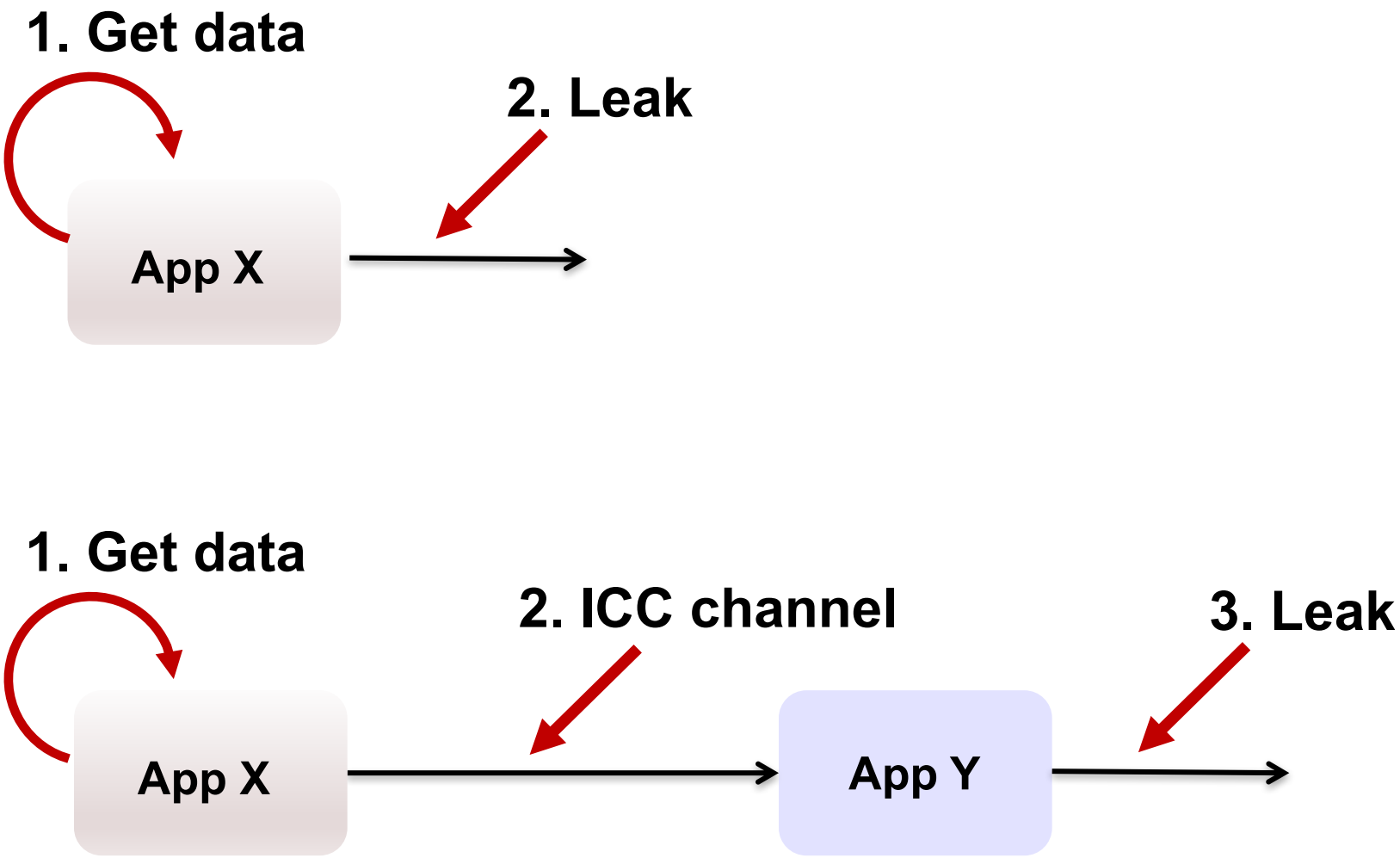
1.8k SHARES

Share on Facebook Share on Twitter +

WHAT'S THIS?

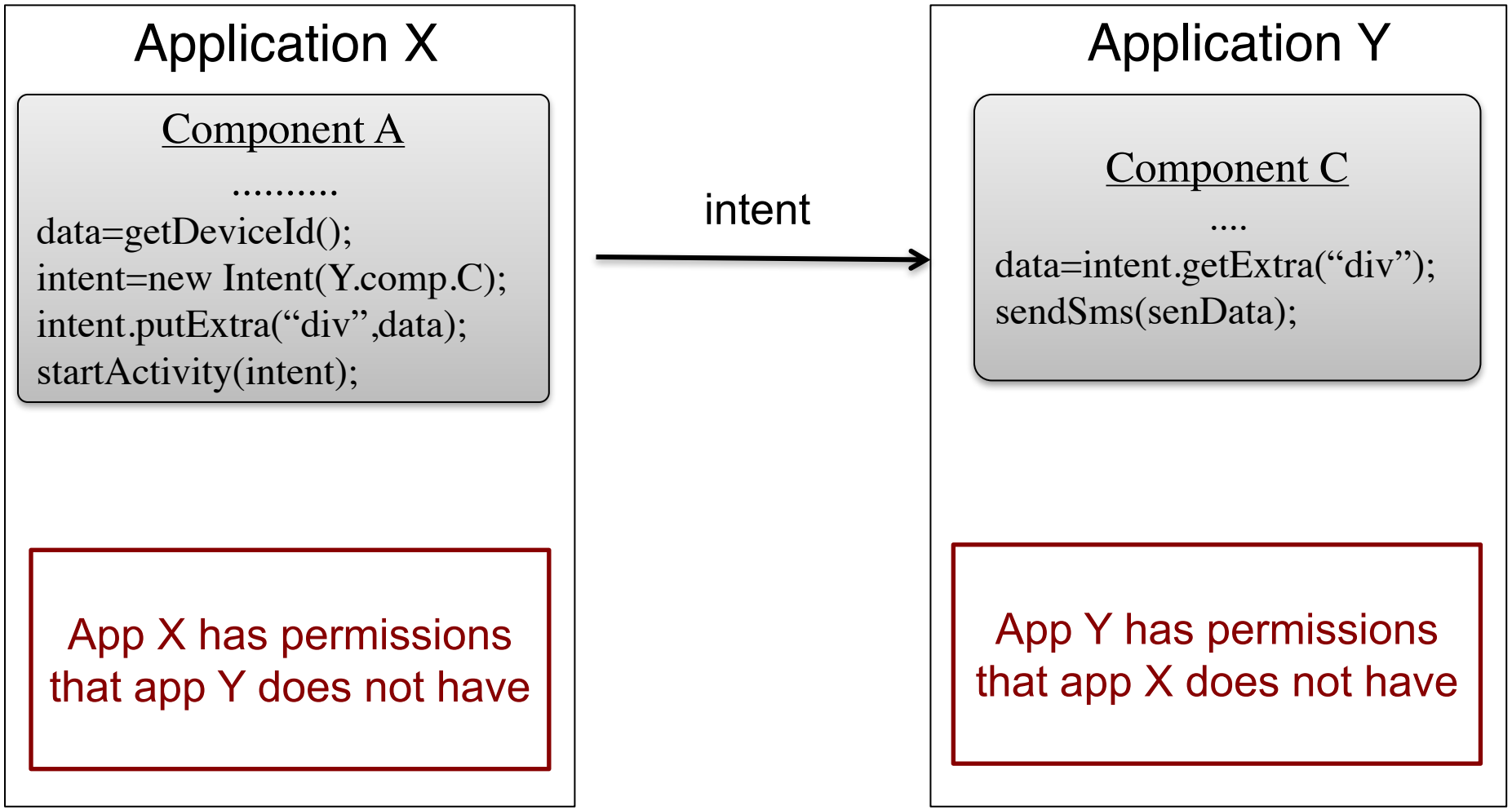
http://mashable.com/2013/10/30/department-of-defense-app-store/#iJuBpfyLJaq4
https://thestack.com/security/2015/02/27

Malware Evolution



[Chin ‘09] [Bugeil ‘11] [Davi ‘11] [Marforio ‘12] [Sbirlea ‘13] [Klieber ‘14] [Bagheri ‘15]

ICC-based Android App Collusion



Android app components

Inter-Component Communication (ICC) via intent

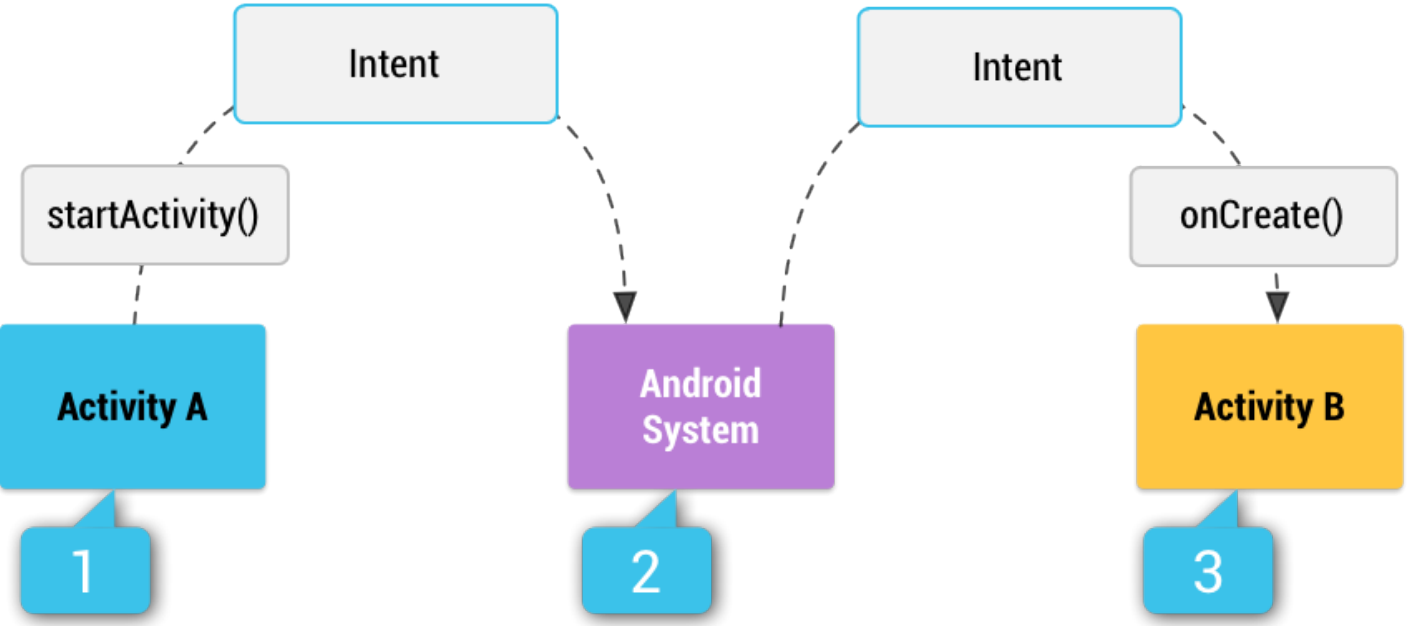
Single screen UI

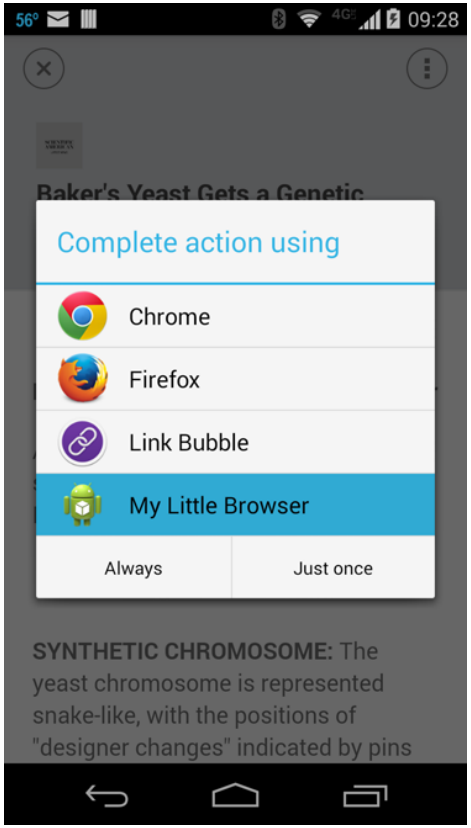
Runs in background

Responds to system-wide announcements

Manages a shared set of app data

Intent Resolution





Implicit / Explicit intents

Explicit Intent

```
intent=new Intent();
intent.setComponent("Y.comp.C");
intent.putExtra(data);
```

Implicit Intent

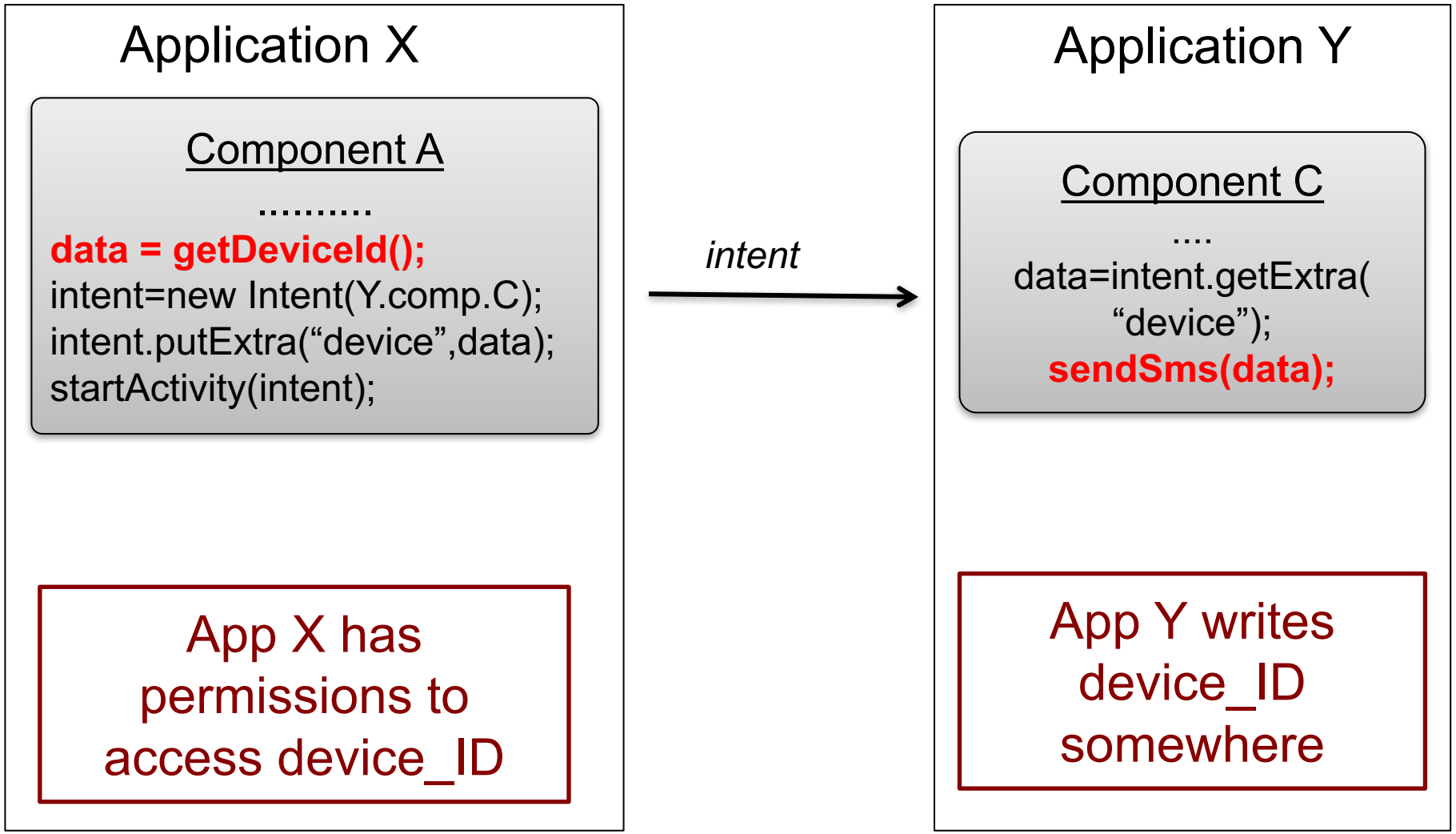
```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.setCategory("android.intent.category.DEFAULT");
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

Who can handle an intent?

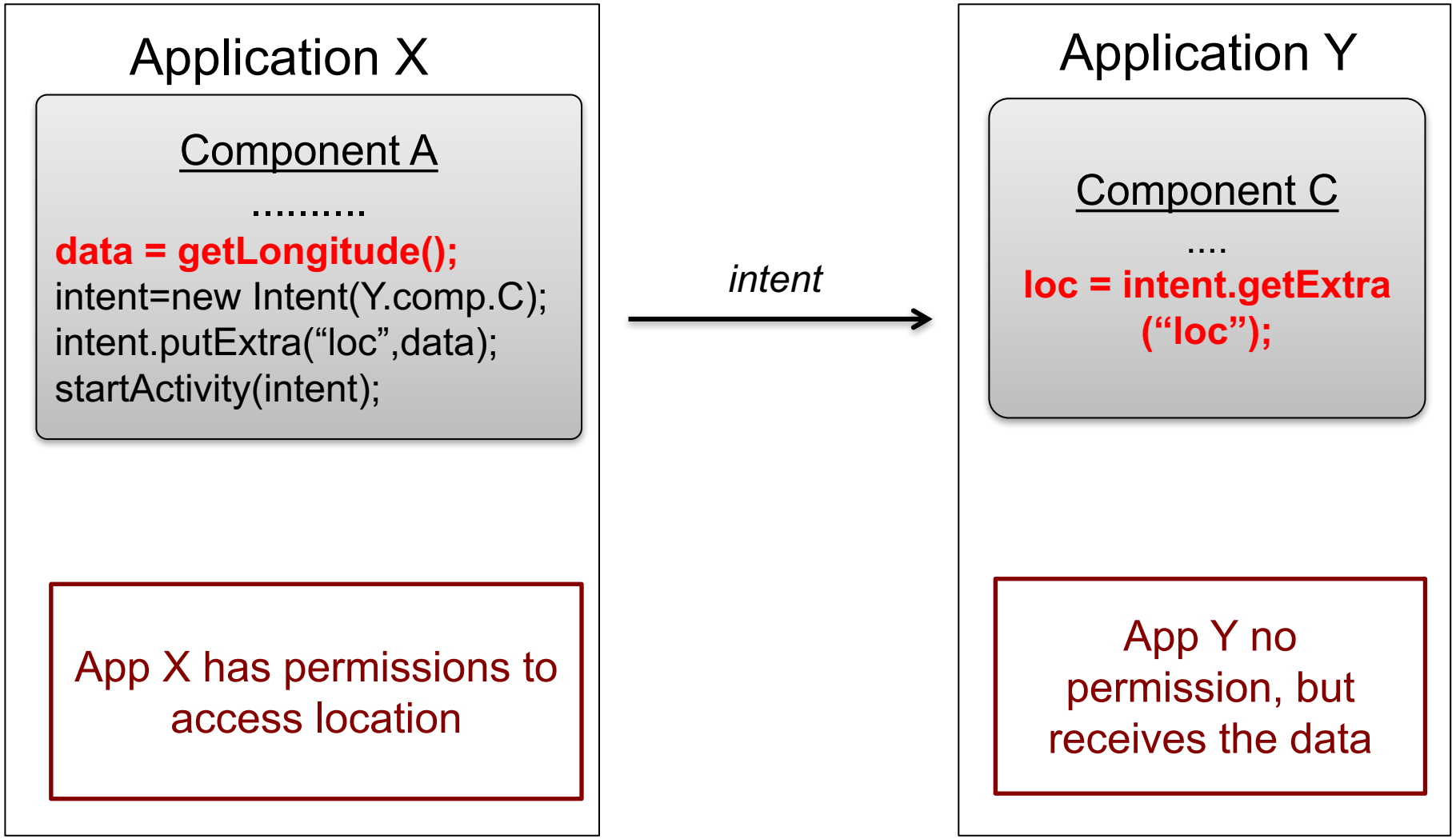
Declared in *AndroidManifest.xml*

```
<activity
android:name="ShareActivity">
    <intent-filter>
        <action
android:name="android.intent.action.SEND"/>
        <category
android:name="android.intent.category.DEFAULT"/>
        <data
android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Threat 1: Collusive data leak



Threat 2: Privilege escalation



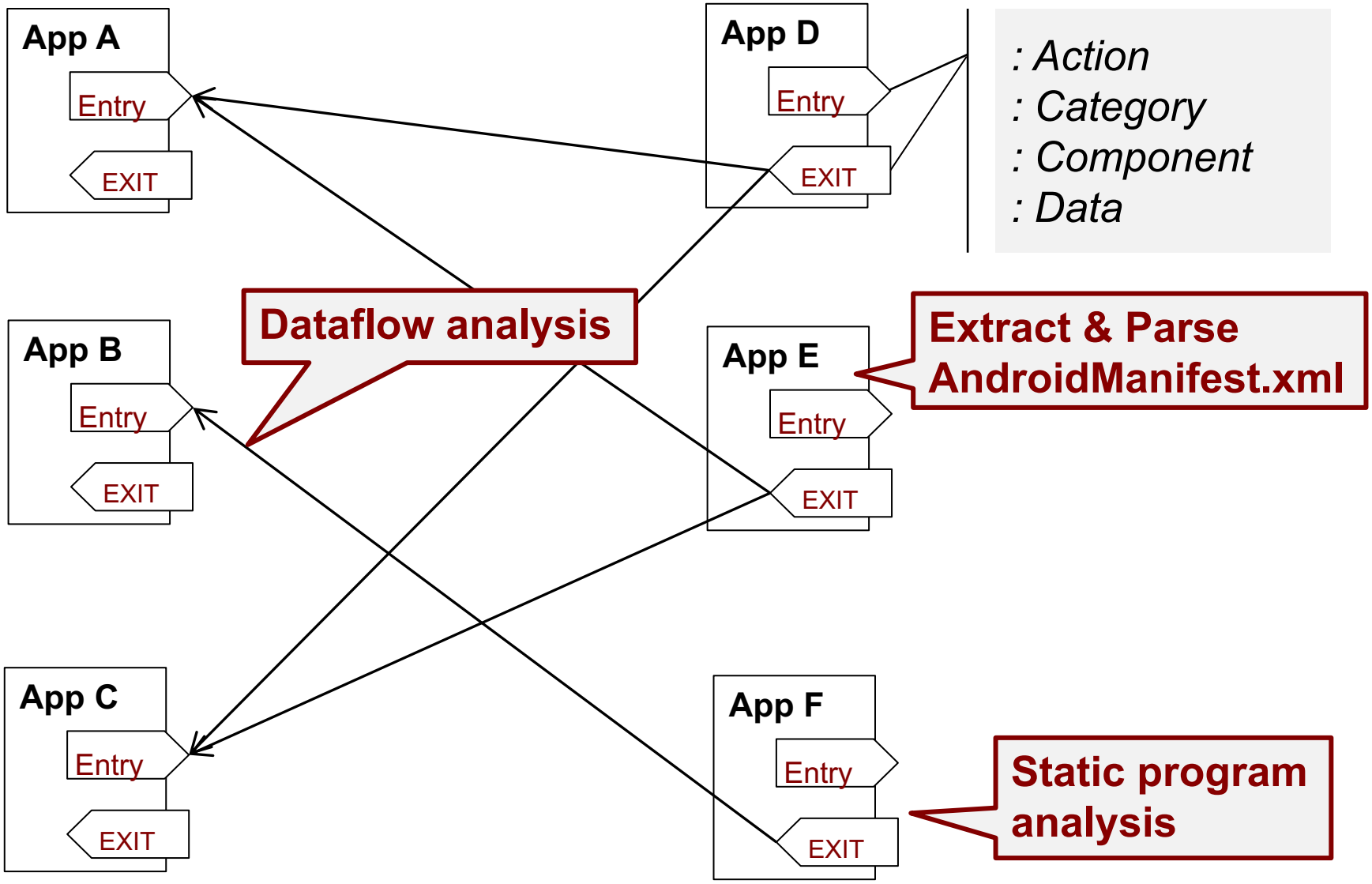
Key challenges

1. N*(N-1)/2 pairs in the worst case
2. Accurate identification of intent fields
3. Flow-level program analysis

- **High precise configuration**
  - Context-sensitive
  - Build complete taint paths
- **Low precise configuration**
  - Context-insensitive
  - Identifies source and sink, not building taint paths
  - May cause false positives

10.7% apps analyzed in low-precise configurations

Overview of our approach



IC3-DIALDroid

- ✓ Can work directly on apk
- ✓ Bug fixes
- ✓ More precise lifecycle modeling
- ✓ Based on IC3

IC3

- Cannot directly work on APK files, needs Dare
- Buggy
- Incomplete lifecycle modeling

	DroidBench			1,000 apps		
	Failed	**# intents**	**Time**	**Failed**	**# intents**	**Time**
IC3	0	27	151s	123	30,640	43hrs
IC3-DIALDroid	0	27	138s	83	39,080	48hrs

-33%

+28%

Dataset statistics (key tables)

Table name	Number of Rows
Classes	3,125,305
Intents	3,294,473
IntentFilters	3,434,119
IntentActions	2,304,744
IntentCategories	210,174
IntentData	1,359,745
ExitPoints	961,960
ICCExitLeaks	52,412
ICCEntryLeaks	249,119
UsesPermissions	839,628
Uris	625,420
Providers	21,405

Sample table: ICCExitLeaks

id	exit_point_id	leak_source	leak_path	leak_sink	method
18	349	$r5 = virtualinvoke $r4. <android.telephony.Telepho...	$r5 = virtualinvoke $r4. <android.telephony.Telepho...	virtualinvoke $r0. <edu.mit.icc_action_string_opera...	getDeviceId
32	835	$r7 = virtualinvoke $r3. <android.location.Location...	$r7 = virtualinvoke $r3. <android.location.Location...	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLastKnownLocati
33	835	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLongitude
34	835	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLatitude
35	836	$r7 = virtualinvoke $r3. <android.location.Location...	$r7 = virtualinvoke $r3. <android.location.Location...	virtualinvoke $r2. <com.tapsistemas.avcanquake.Main...	getLastKnownLocati
36	836	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r2. <com.tapsistemas.avcanquake.Main...	getLongitude
37	836	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r2. <com.tapsistemas.avcanquake.Main...	getLatitude
38	837	$r7 = virtualinvoke $r3. <android.location.Location...	$r7 = virtualinvoke $r3. <android.loca	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLastKnownLocati
39	837	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLongitude
40	837	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r4. <com.tapsistemas.avcanquake.Main...	getLatitude
41	838	$r7 = virtualinvoke $r3. <android.location.Location...	$r7 = virtualinvoke $r3. <android.location.Location...	virtualinvoke $r2. <com.tapsistemas.avcanquake.Main...	getLastKnownLocati
42	838	$d0 = virtualinvoke $r7. <android.location.Location...	$d0 = virtualinvoke $r7. <android.location.Location...	virtualinvoke $r2. <com.tapsistemas.avcanquake.Main...	getLongitude

Original length 922

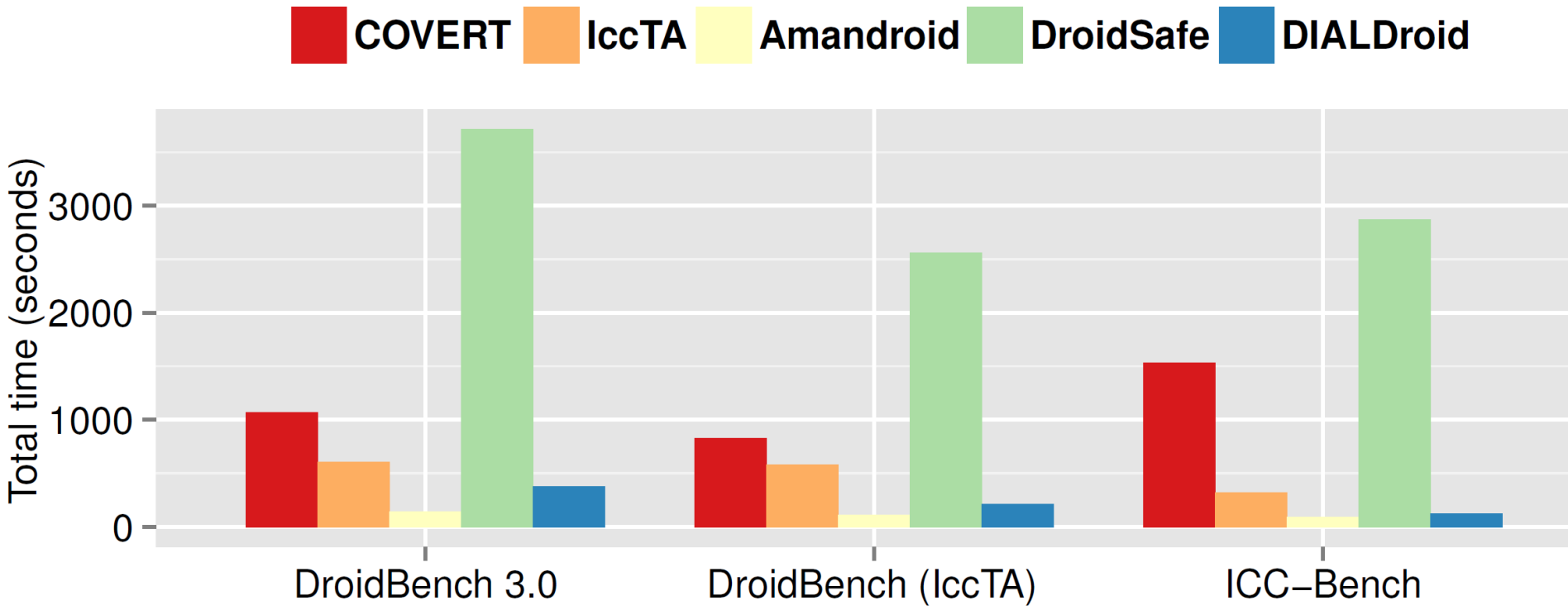
Benchmark Evaluation for Inter-App ICC Performance

Benchmarks used:

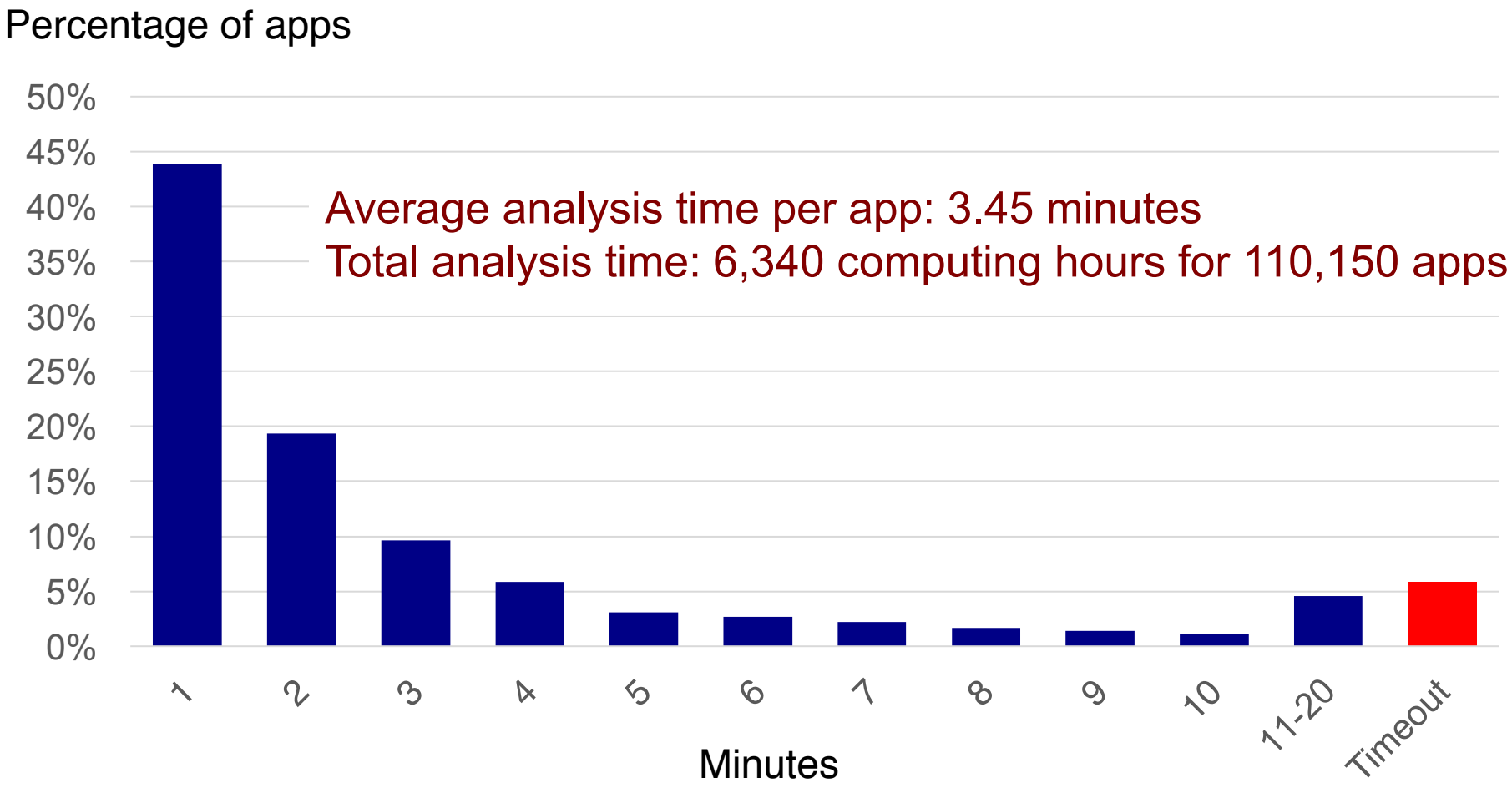
- DroidBench 3.0,
- IccBench,
- DroidBench-IccTA

	COVERT	IccTA	DIALDroid
Precision	3.3%	100.0%	100%
Recall	45.8%	12.5%	91.2%
F-measure	0.06	0.22	0.95

Execution Time on Benchmarks



Analysis time



Results Summary

Threat type	Collusion	Privilege escalation	Intent type	# source apps	#receiver apps	#sensitive ICC channels	Total app pairs
I	Yes	Yes	Explicit	0	0	0	0
II	No	Yes	Explicit	0	0	0	0
III	Yes	No	Explicit	0	0	0	0
IV	Yes	Yes	Implicit	33	1,792	77,104	16,712
V	No	Yes	Implicit	62	44,514	1,785,102	1,032,321
VI	Yes	No	Implicit	21	1,040	34,745	6,783

Among the apps downloaded from google play

Malicious or accidental data leak – that is the question

Case study 1: Same developer privilege escalation

Threat TYPE V [escalation w/o collusive data leak]

com.nextag.android **to** *com.thingbuzz*

- By NexTag Mobile
- *com.nextag.android*
  - retrieves location, sends via an implicit intent
  - compares price across different e-commerce sites
- *com.thingbuzz*
  - accepts the above intent, but has no location permission
  - provides shopping advice to users

Case study: 2

Threat TYPE IV [escalation w/ collusive data leak]

com.ppgps.lite **to** *de.ub0r.android.websms*

- *com.ppgps.lite*
  - retrieves location and sends via an implicit intent
  - provides real-time flight info to pilots of paragliders
- *de.ub0r.android.websms*
  - leaks it via SMS to a phone number
  - has no location permission

Case study: 3

Threat TYPE VI [collusive data leak w/o escalation]

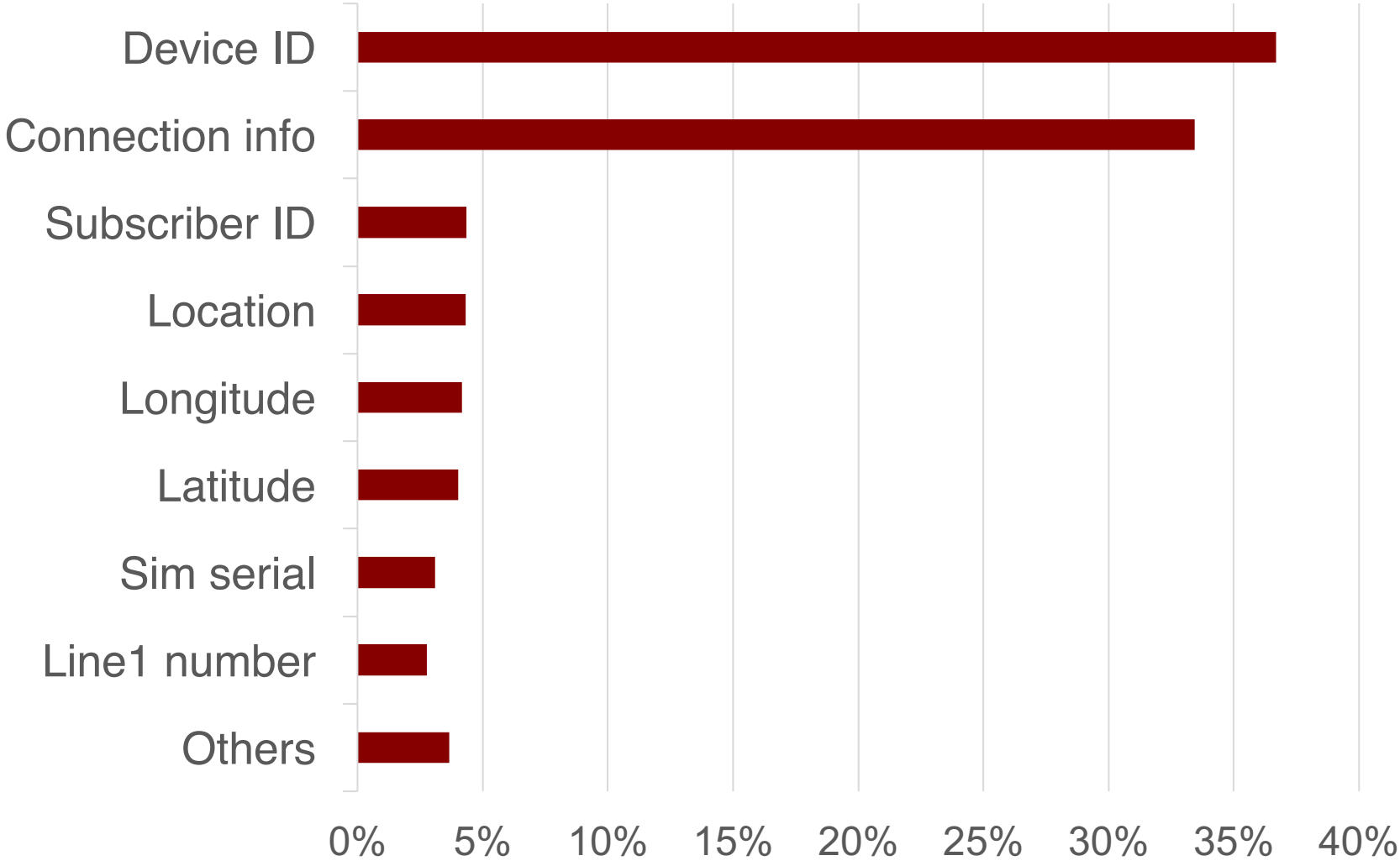
com.ccmass.fotoalbumgpslite **to** *com.ventricake.retrica*

- *com.ccmass.fotoalbumgpslite*
  - retrieves location (getLatitude, getLongtitude)
  - organizes photos based on locations of photos
- *com.ventricake.retrica*
  - accept the above intent, but has location permission
  - writes the data to a log
  - takes photos with various filters

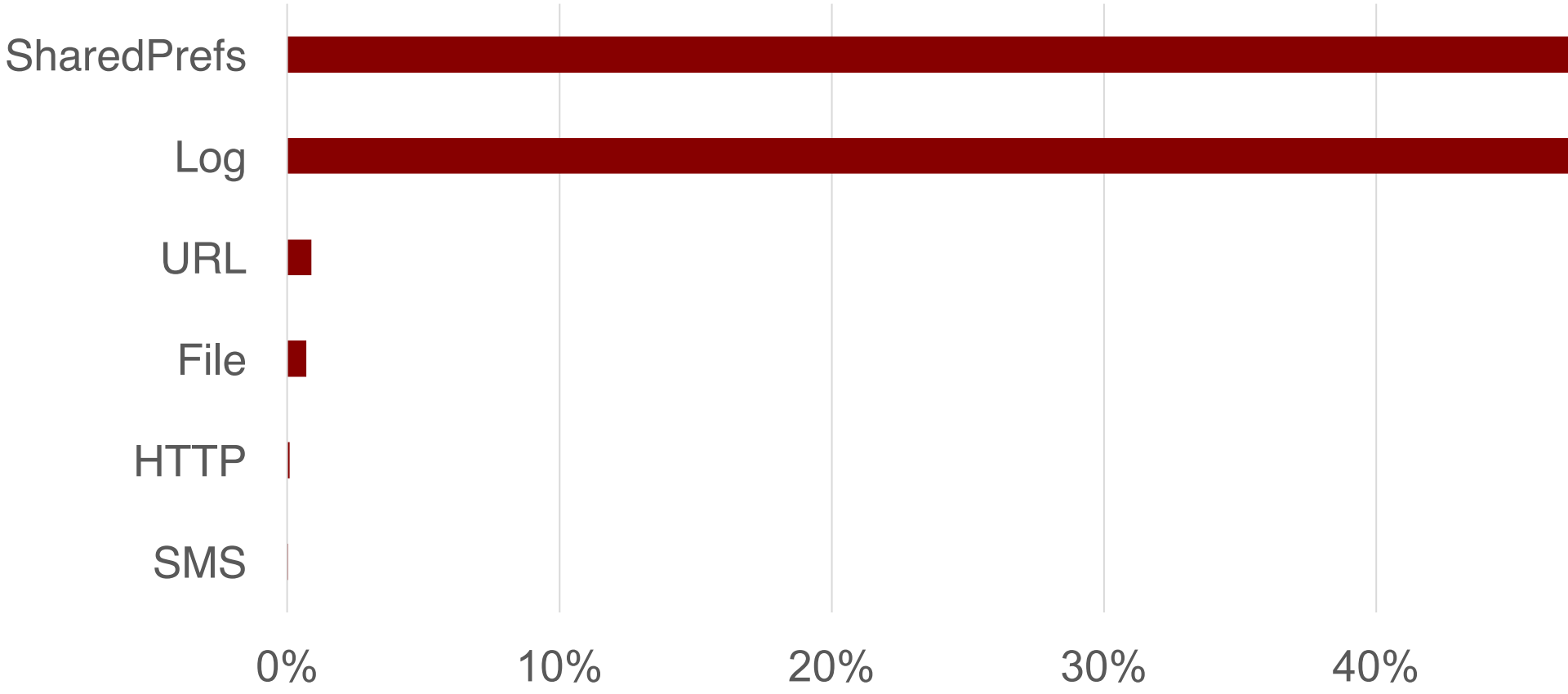
Permission leaks via privilege escalations

Permission	Count
android.permission.ACCESS_FINE_LOCATION	1,155,301
android.permission.ACCESS_COARSE_LOCATION	1,163,769
android.permission.READ_PHONE_STATE	880,645
android.permission.ACCESS_WIFI_STATE	433,887
android.permission.ACCESS_NETWORK_STATE	486
android.permission.BLUETOOTH	153

Distribution of Collusive sources

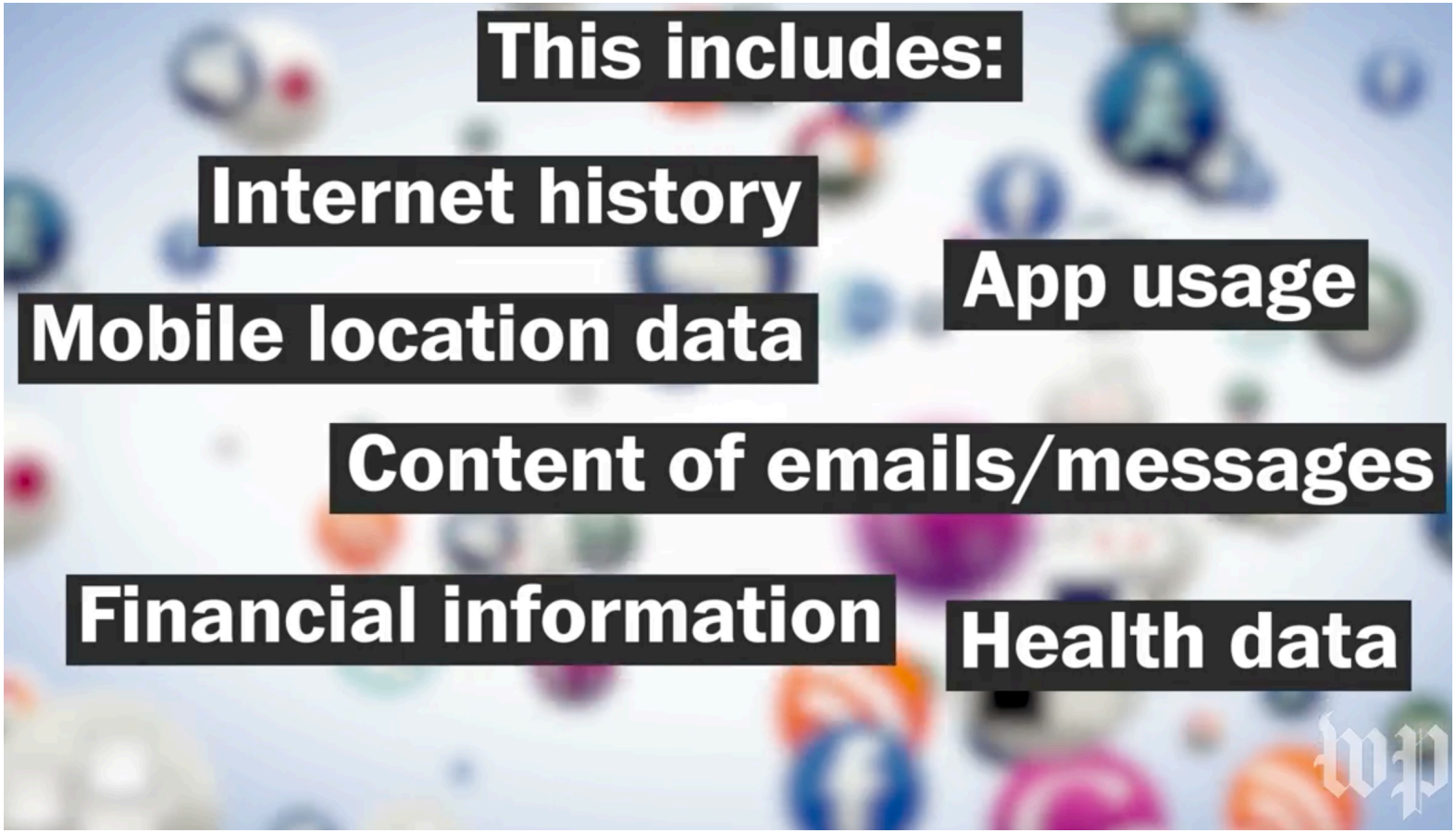


Distribution of Collusive sinks



Privacy, is it a lost battle (at least in US)?

- US Internet service providers (ISP) to monitor customers' behavior online
- without users' permission,
- to use personal information to sell highly targeted ads



[Washington Post, March 28, 2017]

Summary and Open Source

- 110,150 apps analyzed, 0.034% of ICC links carry sensitive info
- No explicit intent based collusion
- device_ID and location leaked the most
- 23,495 colluding pairs among Google Play, originated from 54 apps
- Same-developer privilege escalation involving location

Open source contribution: improved ICC analysis more accurate than state-of-the-arts

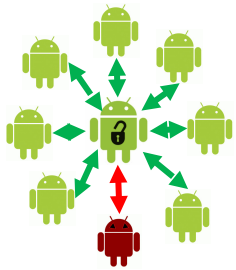
Code and benchmark available: https://github.com/dialdroid-android

Dataset available: http://amiangshu.com/dialdroid

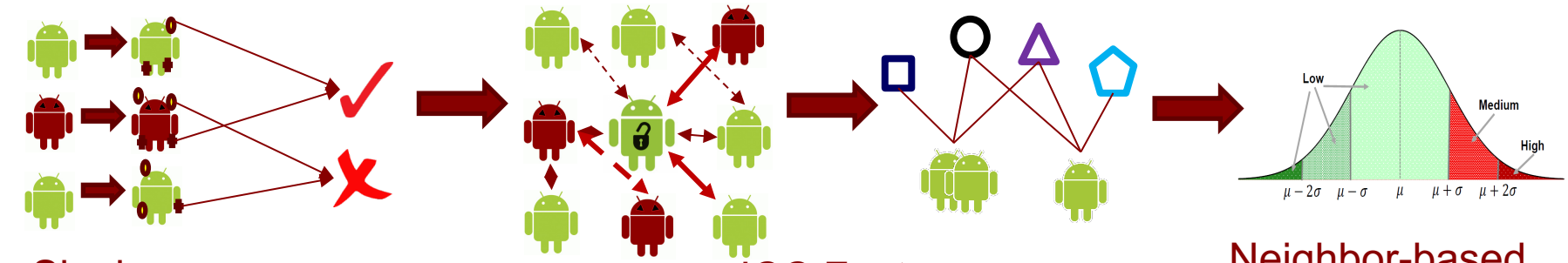
Another Android ICC Work in MoST Workshop in May

IEEE S&P MoST 2017

Prioritize ICC risks based on communication graphs



Distributed MapReduce ICC mapping



Single-app Static Analysis

MapReduce

ICC Feature Extraction

Neighbor-based Risk Analysis

Questions?

Thank you for your attention!