Benchmarking Android Security Analysis

A Bachelors Project,

Final Presentation

by Timo Spring

Supervised by Claudio Corrodi

1. Project Overview

What is it about?

Problem

- Millions of android apps
- Hundreds of analysis tools
- Large scale taxonomies classifying them
- Lack of comparison in practice

Project Idea

- Run selected tools on common dataset
- Compare the results from the different tools

1. Project Overview

Benchmarking concept

Small scale qualitative

- DroidBench dataset (119 apps)
- Common Configuration
- Manually check the validity of the reported leaks

Large scale quantitative

- F-Droid dataset (~1.5k apps)
- Automatically analyse number of detections and matchings

2. Tool Selection Process

Focus on vulnerability detection

ADDICTED, Amandroid, ApkCombiner, App-Ray, AppAudit, AppCaulk, AppCracker, AppFence, AppGuard, AppProfiler, AppSealer, Aquifer, ASM, AuthDroid, Bagheri, Bartel, Bartsch, Bifocals, Buhov, Buzzer, CMA, CoChecker, ComDroid, ConDroid, ContentScope, Cooley, COPES, COVERT, CredMiner, CRePE, CryptoLint, Desnos, DexDiff, DroidAlarm, DroidChecker, DroidCIA, DroidGuard, DroidRay, Droidsearch, Enck, Epicc, FineDroid, Flowdroid, Gallo, Geneiatakis, Grab'nRun, Harehunter, HornDroid, IccTA, IPCInspection, IVDroid, Juxtapp, Kantola, KLD, Lintent, Lu, MalloDroid, Matsumoto, Mutchler, NoFrak, NoInjection, Onwuzurike, PaddyFrog, PatchDroid, PCLeaks, PermCheckTool, PermissionFlow, Poeplau, Pscout, QUIRE, Ren, SADroid, SCanDroid, Scoria, SecUP, SEFA, Smith, SMV-HUNTER, STAMBA, Stowaway, SUPOR, TongxinLi, Vecchiato, VetDroid, WeChecker, Woodpecker, Zuo

2. Tool Selection Process

… only few tools obtainable and runnable

ADDICTED, Amandroid, ApkCombiner, App-Ray, AppAudit, AppCaulk, AppCracker, AppFence, AppGuard, AppProfiler, AppSealer, Aquifer, ASM, AuthDroid, Bagheri, Bartel, Bartsch, Bifocals, Buhov, Buzzer, CMA, CoChecker, ComDroid, ConDroid, ContentScope, Cooley, COPES, COVERT, CredMiner, CRePE, CryptoLint, Desnos, DexDiff, DroidAlarm, DroidChecker, DroidCIA, DroidGuard, DroidRay, Droidsearch, Enck, Epicc, FineDroid, Flowdroid, Gallo, Geneiatakis, Grab'nRun, Harehunter, HornDroid, IccTA, IPCInspection, IVDroid, Juxtapp, Kantola, KLD, Lintent, Lu, MalloDroid, Matsumoto, Mutchler, NoFrak, NoInjection, Onwuzurike, PaddyFrog, PatchDroid, PCLeaks, PermCheckTool, PermissionFlow, Poeplau, Pscout, QUIRE, Ren, SADroid, SCanDroid, Scoria, SecUP, SEFA, Smith, SMV-HUNTER, STAMBA, Stowaway, SUPOR, TongxinLi, Vecchiato, VetDroid, WeChecker, Woodpecker, Zuo

3. Selected Tools In A Nutshell

Tools in a nutshell – pretty much the same

	COVERT	Flowdroid	IccTA	IC3 (Epicc)	Horndroid
Type:	Static & Formal	Static	Static	Static	Static & Formal
Artefact:	Manifest Code	Manifest Layout Code	Manifest Layout Code	Manifest	 Code
Sensitivity	Flow Context	Flow Field Context	Flow Field Context	Flow Field Context	Flow* Field* Context
Sources and Sinks	yes	yes	yes	no	yes
Uses	Flowdroid		Flowdroid IC3	Flowdroid	

* partially

3. Benchmarking Implementation

Runs tools and parses output

Class:	`org.cert.sendsms.ButtonListener`
Method:	`onClick(android.view.View) void`
Sink Method:	`sendMessage(String uid) void`
Detected by:	`flowdroid, iccta`

- Easy to extend with new tools (artefact, parser, results)
- Usability

4. Small Scale Analysis

DroidBench facilitates analysis for true/false positives

- 119 apps with known data leak vulnerabilities
- 125 leaks (sinks) – indicated in source code
- Enables analysis for true/false positives

4. Evaluation – Small Scale Analysis

Metrics for comparison

Number of reported vulnerabilities

→ True / false positives

Precision & recall

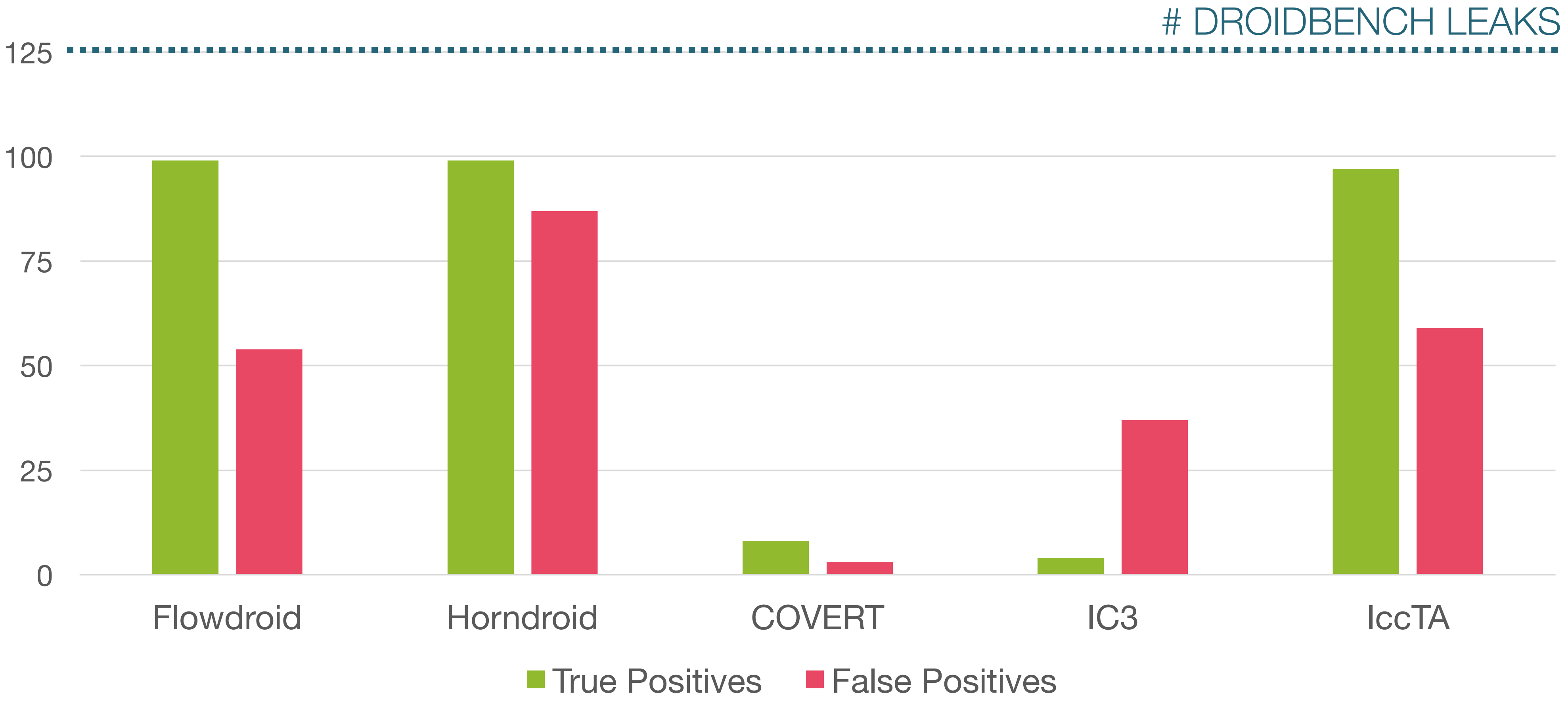
→ Compare performance

McNamar's Test

→ Pairwise comparison (similarity)

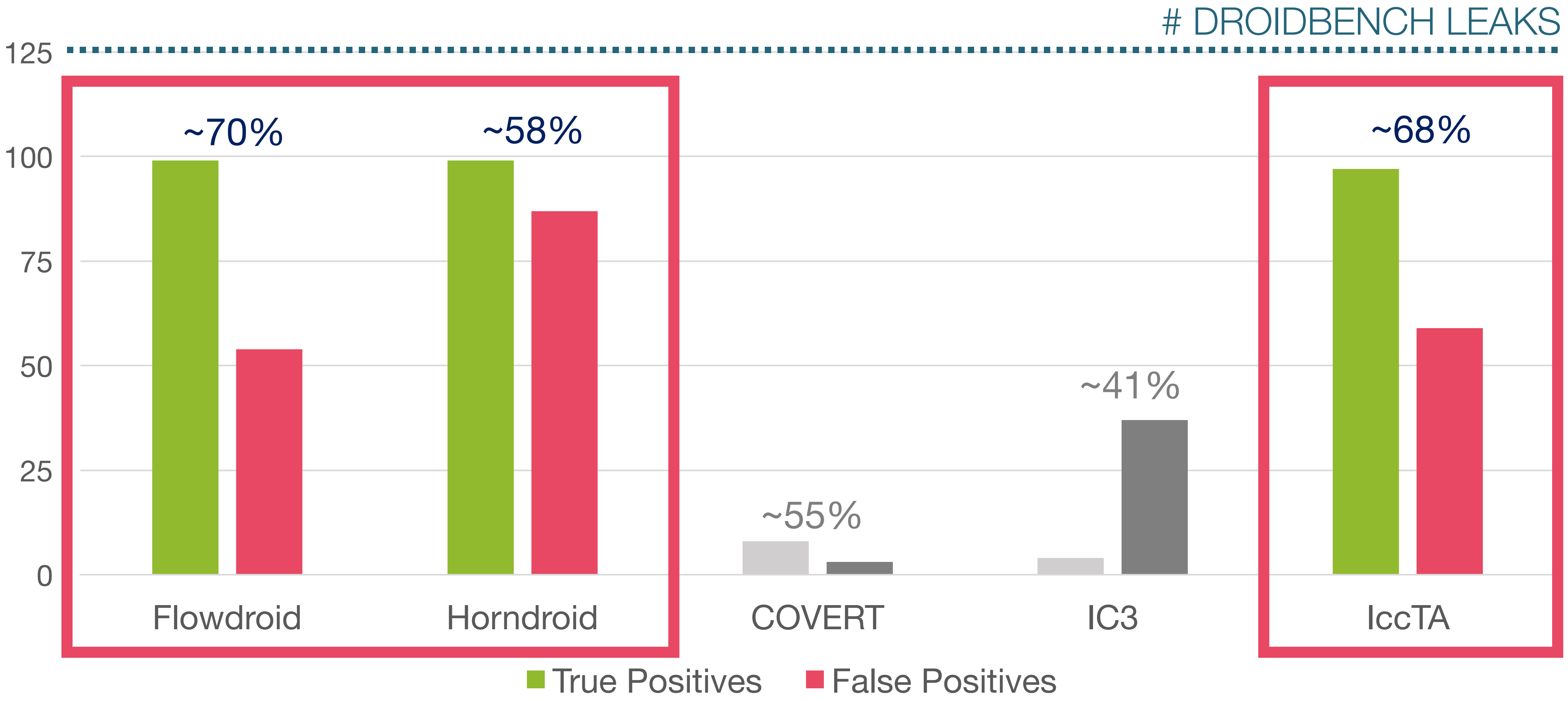
4. Evaluation – Small Scale Analysis

Overview of true and false positives



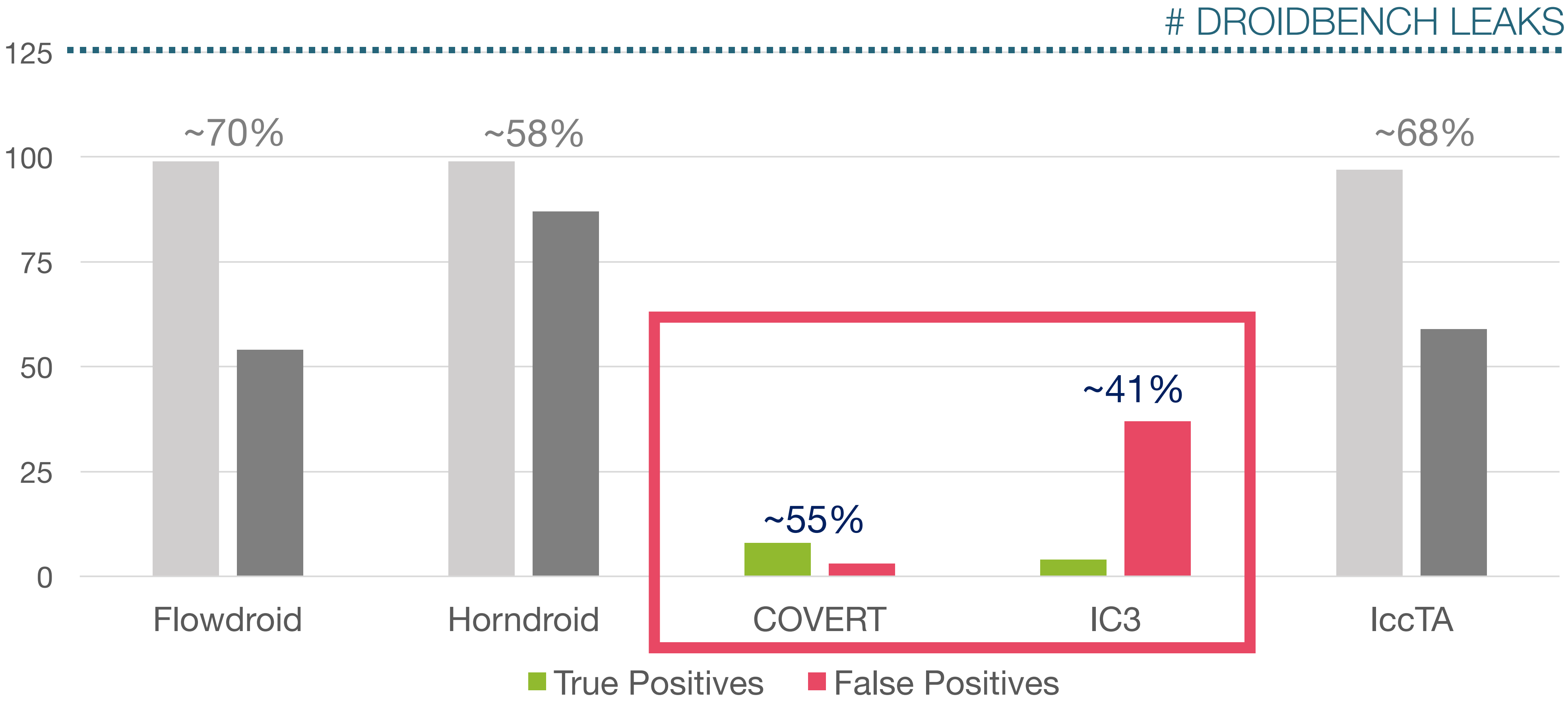
4. Evaluation – Small Scale Analysis

Flowdroid with highest accuracy



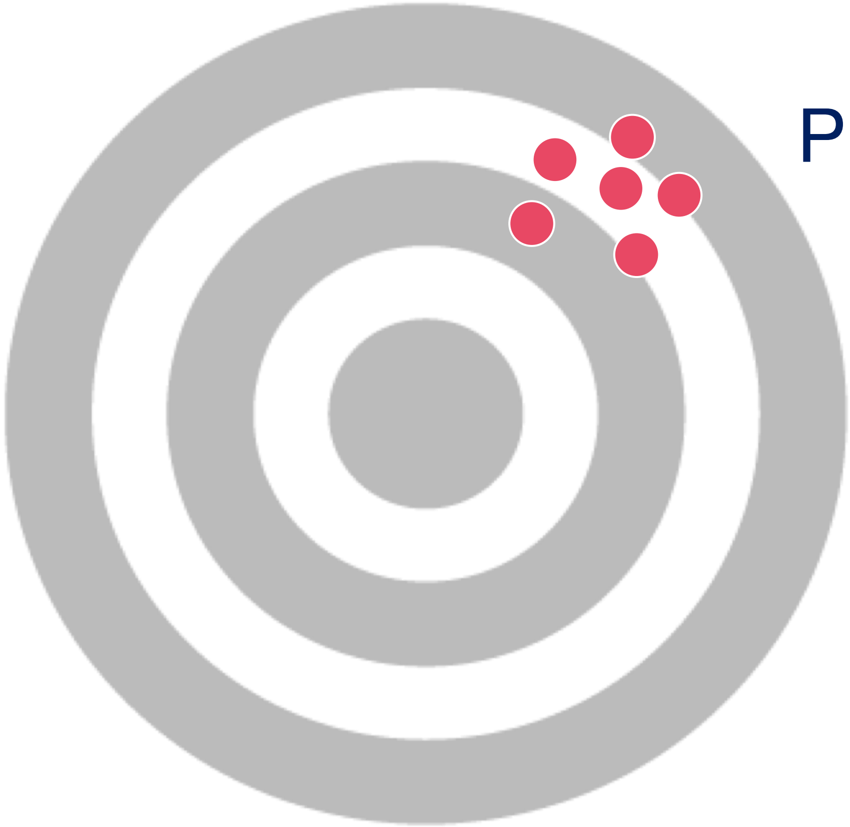
4. Evaluation – Small Scale Analysis

COVERT and IC3 under-perform



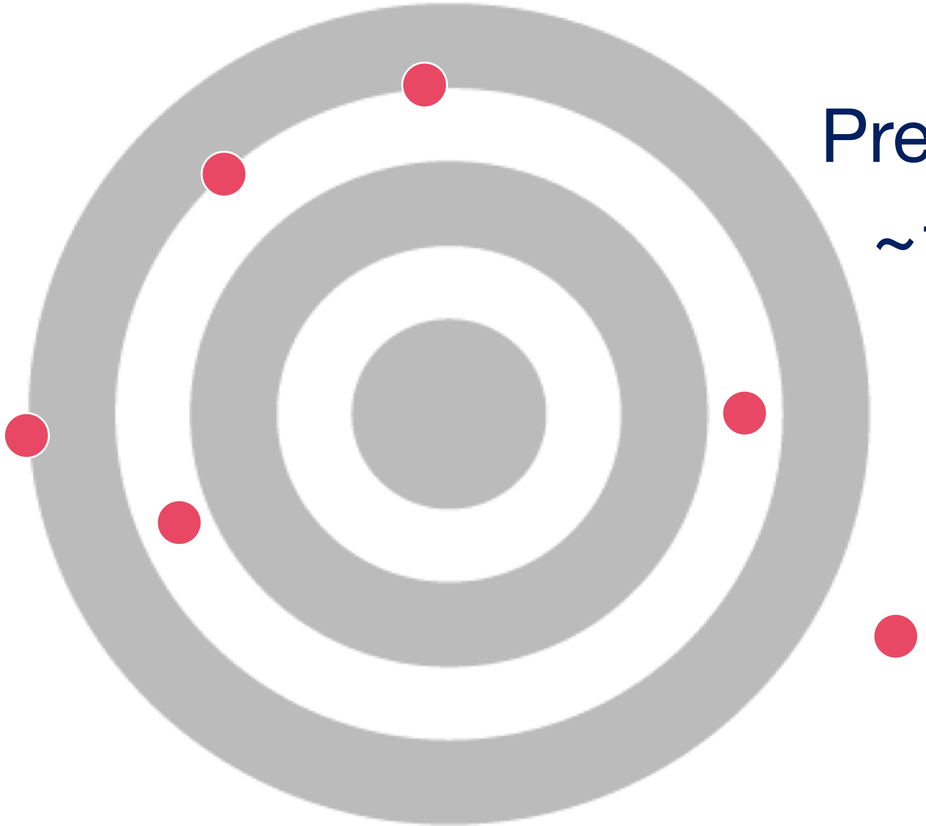
4. Evaluation – Small Scale Analysis

COVERT with highest precision



4. Evaluation – Small Scale Analysis

IC3 very unprecise and inaccurate

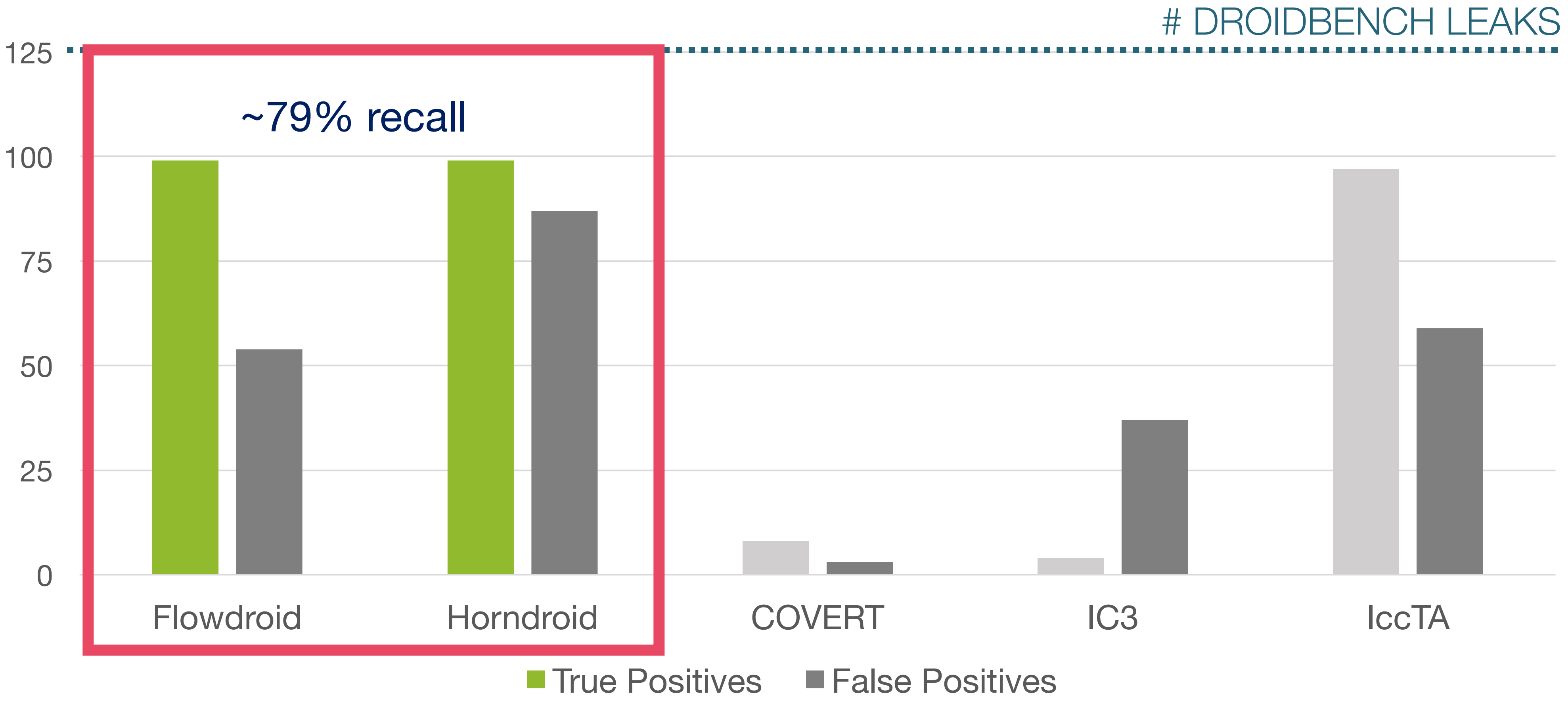


Precision

~10%

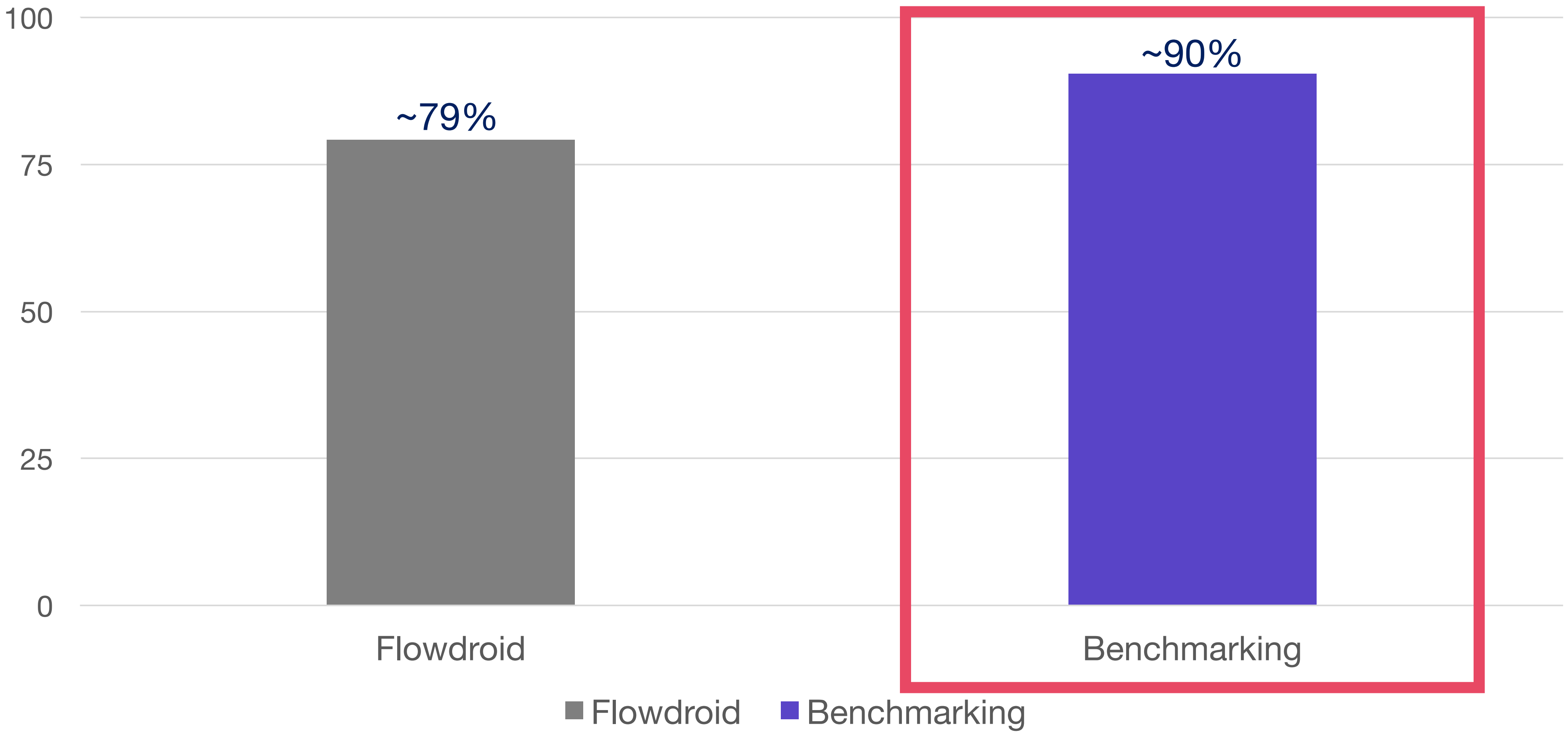
4. Evaluation – Small Scale Analysis

Flowdroid and Horndroid recall most true positives



4. Evaluation – Small Scale Analysis

How about our implementation?



4. Evaluation – Small Scale Analysis

Agreement effect on probability of correct classification

	HornDroid	*COVERT*	*IC3*	*IccTA*
FlowDroid	0.729	0.776	0.653	0.699
HornDroid		0.696	0.476	0.709
COVERT			0.479	0.754
IC3				0.62

Agreement of tools fairly impacts the probability of true classification

4. Evaluation – Small Scale Analysis

Best performing tools are significantly similar

	HornDroid	*COVERT*	*IC3*	*IccTA*
FlowDroid	9.94	10.56	35.29	2.77
HornDroid		0.2	8.53	6.01
COVERT			26.33	6.97
IC3				28.99

Statistical significant similarities among tools are observable

4. Evaluation – Small Scale Analysis

Summarized

- Flowdroid has best performance among tools
- Benchmarking can leverage base approaches
- Tools with better performance tend to be significantly similar

5. Next steps

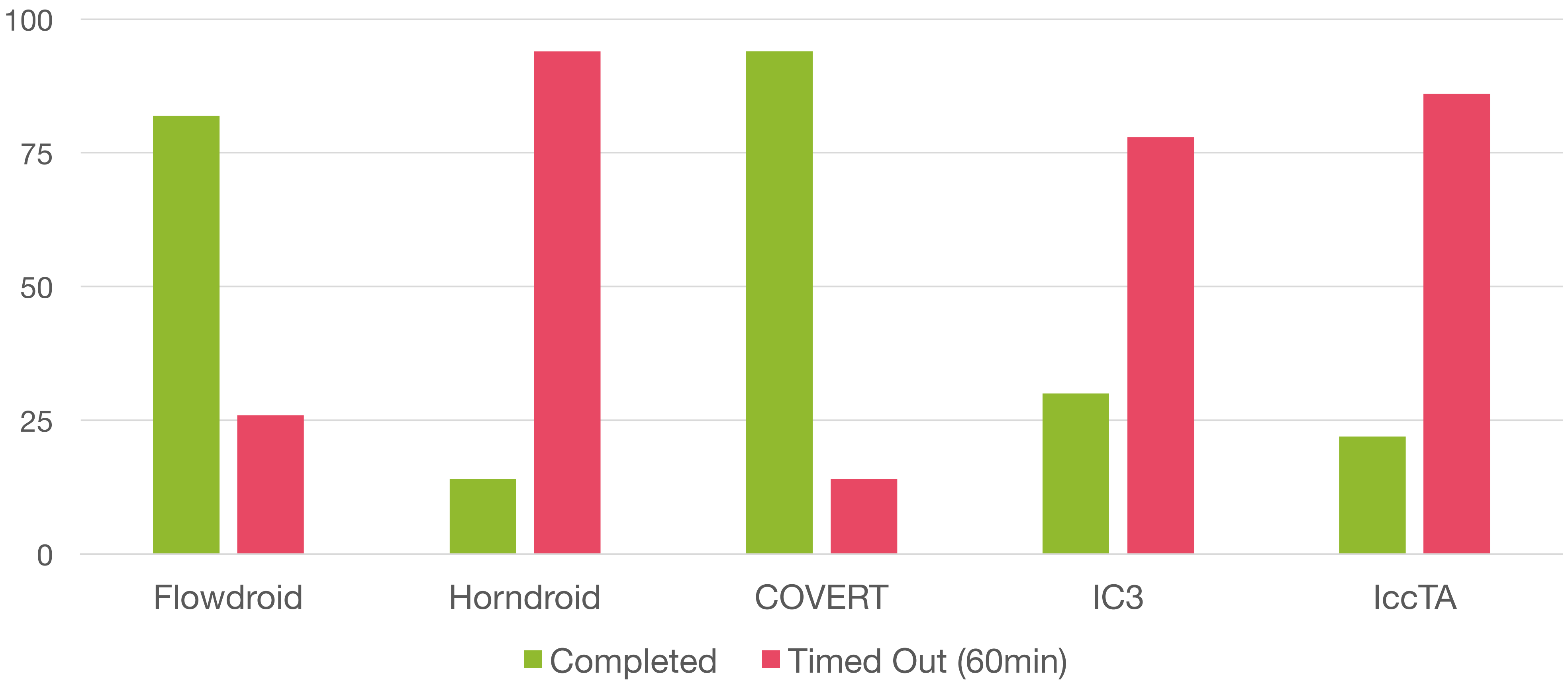
Large Scale Analysis

- Run analysis on F-Droid data set (~1.5k real world apps)
- Verify number of matchings among tools
- Already detected 669 vulnerabilities for 108 real world apps
    - 10 vulnerabilities are reported by at least two tools

Time consuming: A lot of time outs, especially for Horndroid

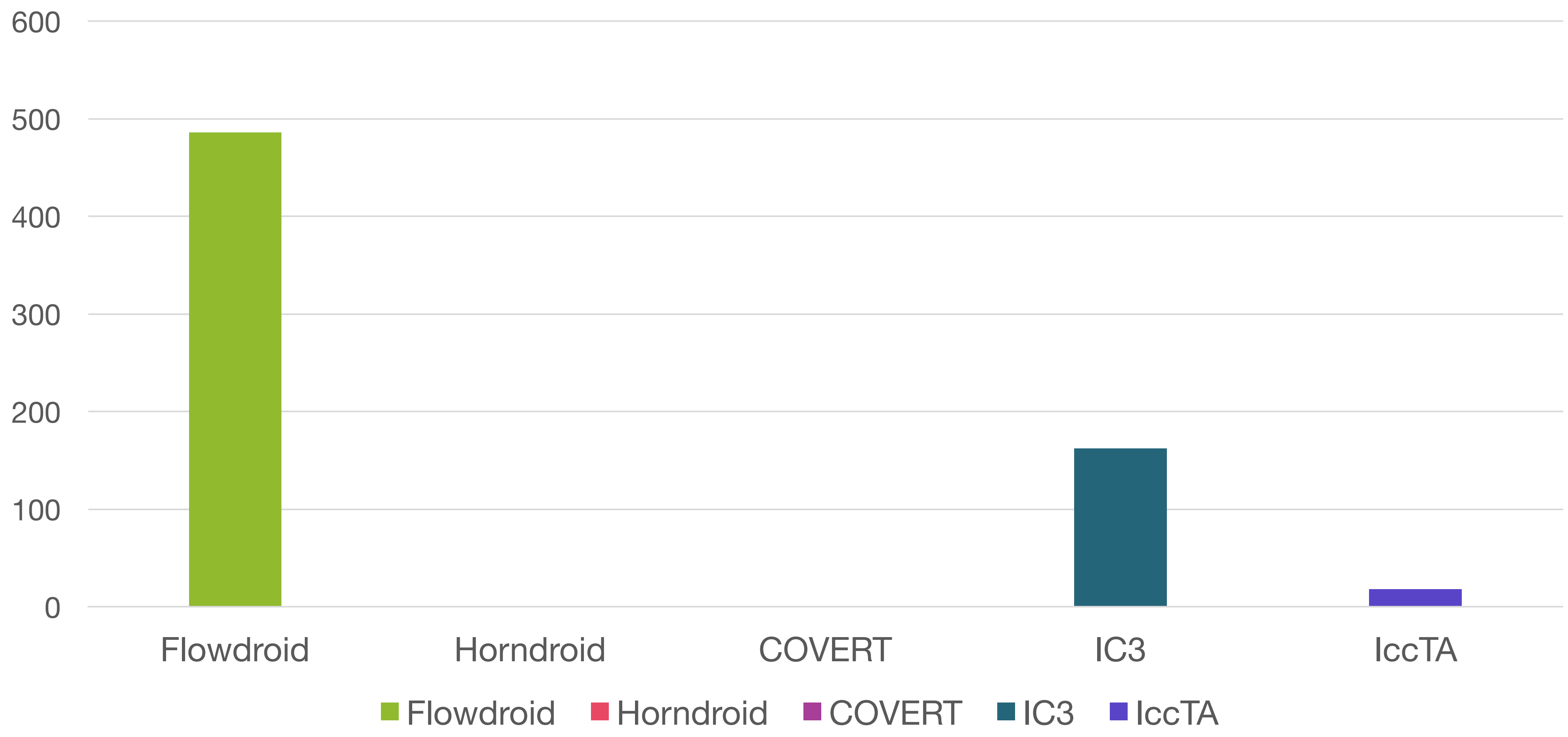
4. Evaluation – Large Scale Analysis

Analysis are time consuming



4. Evaluation – Large Scale Analysis

Data leaks are present in real world applications



6. Lessons Learned

User need for Benchmarking tools

The availability of artefacts in the Android security domain is poor

Similar structure does not mean similar performance

Benchmarking can leverage base approaches and increase quality of results

7. Bonus

Paper submitted to ESSoS18

Engineering Secure Software and Systems

Benchmarking Android Data Leak Detection Tools

Claudio Corrodi, Timo Spring, Mohammad Ghafari, and Oscar Nierstrasz

Software Composition Group, University of Bern, Bern, Switzerland

Abstract Security of mobile application available in virtual stores is a concern because platform providers cannot vet every published application. Consequently, many applications—both malign and benign—exhibit security issues, such as leaking of sensitive data. In recent years, researchers have proposed a myriad of techniques and tools to detect such issues. However, it is unclear how these approaches perform compared to each other. The tools are often no longer available, thus comparing different approaches is almost infeasible.
In this work, we review approaches for detecting data leaks in Android applications. From an initial list of 87 approaches, only 5 could be obtained and executed, and produced results in the selected domain. We compare these using a set of known vulnerabilities and discuss the overall performance of the tools.
We further propose an approach to compare security analysis tools by normalising their interfaces, which simplifies result reproduction and extension.

Keywords: data leak, Android, benchmarking

1 Introduction

Security of mobile applications is a hot topic in both research and industry. With millions of available applications in virtual stores, platform providers such

Backup

4. Backup – Formulas

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

McNamar's Test:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}$$

Confidence Interval: 99%

4. Backup – Small Scale Analysis

Custom Configuration reduces number of reported leaks