Pattern-based Inter-Component Communication Discovery for Android Applications[[**Marcelo:** consider changing title to stress what it achieves as opposed to how it achieves.]]

Vinicius Souza
Federal University of Pernambuco
Recife, Brazil
vcps@cin.ufpe.br

Leopoldo Teixeira
Federal University of Pernambuco
Recife, Brazil
lmt@cin.ufpe.br

Marcelo d'Amorim
Federal University of Pernambuco
Recife, Brazil
damorim@cin.ufpe.br

ABSTRACT

Android applications are built based on a variety of components. Therefore, to analyze such applications, one usually has to identify inter-component communication (ICC). In this paper, we explore the fact that such communications often follow some patterns. Therefore, we propose a lightweight, pattern-based approach for detecting ICC. We apply this analysis to XX applications from the Google Play store, identifying $YY\%$ of ICC specifications. Although our analysis does not assure soundness (no false negatives) or completeness (no false positives) by definition, our results provide strong evidence that it is competitive to others in terms of precision and recall. Considering efficiency, however, our analysis outperforms existing analysis by XXX. These results show that our approach can be useful for integration into other program analysis tools for Android.

CCS Concepts

•General and reference → Empirical studies; •Software and its engineering → Automated static analysis;

1. INTRODUCTION (1 PAGE)

Android marketshare, numbers, importance. Briefly explaing that Android applications interact through intents and ICC

Discuss past analysis, how they often are built for soundness, and therefore suffer on scalability. Also mention about trying to capture each and every way of building intents and ICC

Explain that intents and ICC are built following patterns. Use a small example?[[**Marcelo:** Yes, please. This is very very important. You can have one super-short simple-to-describe example here and have a section of example(s) following this intro section.]] We can explore this to create lightweight analysis, that are imprecise, but can account for most of the scenarios, as we show in the evaluation (hopefully).

Brief idea of the solution. Reference the soundness manifesto [5] when explaining it.

We make the following contributions:

- **Idea.** We leverage the idea of overlapping analysis information to reduce cost of static analyses.[[**Marcelo:** check if this is really new. Highlight how this embraces the philosophy of soundness.]]
- **Implementation.** We implemented PBICC, a tool for computing Android ICC in a pattern-based way; [[**Marcelo:** Think about a good name. Maybe PBICC is not the best.]];
- **Evaluation.** We perform a study with XX applications, comparing the results with the state-of-the-art analysis for detecting ICC [2, 4, 7]. We also analyze the impact that our solution has on tools that perform taint analysis on Android applications. [[**Leopoldo:** include results when available]]

2. BACKGROUND (1-2 PAGES)

Quickly explain architecture of Android applications, built around components. Explain that components communicate with each other through Intents, defining what is inter-component communication (ICC). Give an example of intent and communication between Android components.

3. EXISTING SOLUTIONS FOR ICC DISCOVERY

Use numbers on entry/exit points from the Epicc paper [7]: "*About 92% of exit points had a single Intent specification*". [[**Leopoldo:** This provides the intuition that a simpler analysis might suffice in the vast majority of cases]]

Go over existing solutions and explain why they are so expensive [[**Marcelo:** If this is short (say, half a column) I would present this in intro very early on, even before presenting our approach.]]. Is this dependent on the size of the program, or in the number of ICC points?

- epicc: build the call graph amounts to large part of the time, *still getting the grasp of the paper, but the problem seems to be that they model the entire application (therefore causing the graph to be too big), while we are mainly interested on specific points of communication*

FSE '16 Seattle, USA
© 2015 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

- ic3: builds on top of epicc, including the constraint propagation solver for resolving strings and URIs. Therefore, it can only be as fast as epicc is, but not faster; depending on the intent values it will also spend some time resolving the strings

- gator (atanas rountev): still working through the ICSE paper and have also downloaded the tool to figure it out

Characterize patterns for creating intents and establishing ICC. Gather the data collected by Vinicius and present here, that in general, half of the ICCs are explicit, and we can infer such information without sophisticated analysis.

4. SOLUTION (1-2 PAGES)

Explain the idea of patterns using 1-2 concrete examples. (1 from motivation and maybe other here)

The intuition is that patterns can often occur together, therefore we might calculate the intersection and give a precise answer over the ICC specification in most of the program points.

4.1 Explicit ICC

- Intent creation with class name (anonymous object)

- Use setComponent/setClass on Intent object

Both patterns can be specialized when the class name is in a local variable, static class attribute, and so on... Explain this.

4.2 Implicit ICC

- Intent with get/put extras

- Intent with Action, Data, and Category fields

4.3 Dynamic Broadcast Receivers

Still working on which patterns we can establish for this case.

4.4 Implementation

Explain the overall implementation, we collect the information based on the patterns established above.

Then we have to perform a "join" of this information, to make it more precise

This results in a number of ICC specifications, or interfaces. Explain the idea using the 1-2 examples given before.

5. EVALUATION (4 PAGES)

[[**Marcelo:** I suggest to think about the questions we want to answer first: I think it is a good idea to show purpose. How the technique (name) performs in term of precision, recall, and efficiency? What is the practical effect of the technique in a client analysis? (When there is loss, how relevant is it? if not, why?) The interesting question here seems to be: if this is for security, do attackers typically care about creating hard-to-find communication patterns? Would they care after reading our paper? Btw., are there client analysis other than security (e.g., tainting)?]]

5.1 Experimental Setup

[[**Marcelo:** The setup/methodology should go together with the experiment, not separate. Don't need to necessarily show independent/dependent variables etc.]]

Explicitly detail the environment in which we performed all of the analysis detailed below

Discuss in which

5.2 Quantitative Analysis

Analysis on precision/recall of the specifications.

Epicc evaluation consists of inferring specifications automatically, and if there is only "one" possible specification, they consider it precise. One way would be to adopt the same strategy for considering our precision/recall. [[**Marcelo:** I don't understand what exactly "specification" means here. Is it the set of ground-truth component-to-component flows? Even more is to understand how they computed/obtained this set. I can only imagine they did that manually but that would only work for small apps. But if the metric recall is not taken into account then the problem is much easier. As the set of flows our analysis reports may not be outrageous, then it is even possible to inspect the output to measure precision (not recall). In summary, there are two alternatives: (1) model set of flows for every app (to enable computation of precision and recall) and (2) inspect output of tool (to measure precision).]]

I am going through FlowDroid [1] and Atanas' GATOR paper [8], to understand how they considered the ground truth.

5.3 Comparison with existing ICC tools

Compare deltas with Epicc [7], IC3 [6], ComDroid [2][[**Marcelo:** is this another ICC tool? If so, also explain in the beginning.]]

5.4 Application to Vulnerabilities

Use IccTA [4] with our ICC information and compare performance and analysis results.

6. RELATED WORK (1 PAGE)

[[**Marcelo:** discuss alternative techniques to solve the same or similar problem (works related by problem) or use an approach similar in spirit to solve a different problem (works related by solution).]]

Epicc [7]
IC3 [6]
IccTA [4]
FlowDroid [1]
GATOR [8]
William Enck's papers [3] (grab more)

7. CONCLUSIONS (1/2 PAGE)

8. REFERENCES

[1] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 259–269, New York, NY, USA, 2014. ACM.

[2] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 239–252, New York, NY, USA, 2011. ACM.

[3] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[4] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, Y. Le Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 280–291, Piscataway, NJ, USA, 2015. IEEE Press.

[5] B. Livshits, M. Sridharan, Y. Smaragdakis, O. Lhoták, J. N. Amaral, B.-Y. E. Chang, S. Z. Guyer, U. P. Khedker, A. Møller, and D. Vardoulakis. In defense of soundiness: A manifesto. *Commun. ACM*, 58(2):44–46, Jan. 2015.

[6] D. Octeau, D. Luchaup, M. Dering, S. Jha, and P. McDaniel. Composite constant propagation: Application to android inter-component communication analysis. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 77–88, Piscataway, NJ, USA, 2015. IEEE Press.

[7] D. Octeau, P. McDaniel, S. Jha, A. Bartel, E. Bodden, J. Klein, and Y. Le Traon. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 543–558, Berkeley, CA, USA, 2013. USENIX Association.

[8] S. Yang, D. Yan, H. Wu, Y. Wang, and A. Rountev. Static control-flow analysis of user-driven callbacks in android applications. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 89–99, Piscataway, NJ, USA, 2015. IEEE Press.