Cooperative Android App Analysis

Felix Pauck, SonarSource, Suttner-Nobel-Allee 3, 44803 Bochum, Germany

Reference
[4] Felix Pauck. *Cooperative Android App Analysis*. Dissertation, Paderborn University, 2023, DOI: 10.17619/UNIPB/1-1698

Supervisor: Prof. Dr. Heike Wehrheim

Date of Defense: 24.03.2023

Abstract

In this summary, the three main contributions of the thesis "Cooperative Android App Analysis" [4] are presented.

The first contribution proposes the cooperative analysis approach. The centerpiece of this approach is the AQL (Android App Analysis Query Language) – a domain specific query language. It allows formulating (AQL-)queries in order to interact with arbitrary analysis tools. As counterpart AQL-Answer come into play, which are able to universally but well-structured embody any kind of analysis result. The second contribution uses the AQL to define reproducible benchmarks that can be used to automatically evaluate analysis tools on such. Various benchmarks are then used in the third contribution to conduct a thorough evaluation of 13 Android taint analysis tools.

Please note, in the context of the thesis, the cooperative analysis implementation is tailored to Android taint analysis, however, the concept can be applied to any kind of analysis.

Introduction. For a while Android is the most-used mobile operating system. Smartphones using it often deal with sensitive and security-critical information, such as contact data or GPS locations. Hence, this data must be protected against attacks. In this regard, there is an arms race taking place between attackers, trying to find new ways of exfiltrating data, and defenders, who develop novel defensive instruments. One such instrument, that is steadily improved, is program analysis. To keep up in the race, we purpose the concept of *cooperative analysis*, that allows defenders to quickly and precisely react on new analysis challenges.

Cooperative Analysis To build a cooperative analysis we want to interact with arbitrary analysis tools and bring together their (often complementary) features. To intuitively and *briefly* describe the concept, let us take a look at the example depicted in Figure 1. Depicted is one Android app with two components (`Main-` and `TargetActivity`). Both components implement a method (`onCreate(...)`), that is executed once the respective component is triggered. The content of these methods is described next.

The **source** (`getDeviceId()`) statement accesses the IMEI number of an device allowing to identify and track an individual, hence, it should not be leaked. Its counterpart, the **sink** (`sendTextMessage(...)`), allows sending a short message to a potential adversary. In this example, a so-called *taint flow*, that leaks the sensitive information, can be found between these two. It uses *inter-component communication (ICC)* to do so – the data extracted in one component is transferred to another that leaks it.

To build a cooperative (taint) analysis that is able to analyze this scenario, we employ the AQL (Android App Analysis Query Language), a domain specific query language that allows us to interact with arbitrary analysis tools. With the AQL, we can formulate the following exemplary (AQL-)query:

```
Flows IN App('ex.apk') ?
```

Once it is processed, e.g., by the AQL-System [4, 3], we get an AQL-Answer that potentially reveals the taint flow. Therefore, the AQL-System must have been configured with an analysis tool that is able to directly find taint flows that involve ICC. Often such a tool is not available. So let us assume the query is answered by FlowDroid [1], the most advance and broadly used Android taint analysis tool. It gives us the partial taint flows 1. and 3. (see Figure 1). To detect the whole taint flow, we must also identify the ICC flow (2.) between. Hence, we formulate another query:

```
Flows IN App('ex.apk') FEATURING 'ICC' ?
```

This may now trigger a different tool that is able to detect ICC flows. These two queries can now be brought together to build a simple but effective cooperative analysis by asking this query:

```
CONNECT [
    Flows IN App('ex.apk') ?,
    Flows IN App('ex.apk') FEATURING 'ICC' ?
] ?
```

The inner queries are answered as before by two individual analysis tools and the `CONNECT` operator (implemented in the AQL-System) that is surrounding both

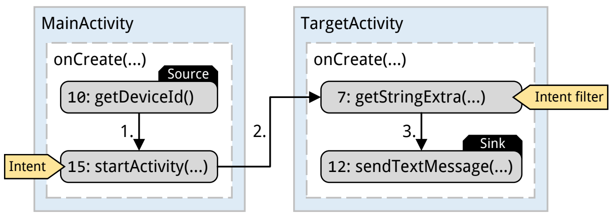

Figure 1: Illustration of the running example

queries brings together the respective AQL-Answers. In our example, it stitches the detected taint flows together with the ICC flow, such that the whole taint flow becomes visible.

This example just provides a glimpse at the possibilities the AQL and the cooperative analysis concept offer. The thesis [4] itself allows taking a look at the full picture.

Reproducible and Automatic Benchmarks. With the AQL at hand, we can formulate the ground truth of a benchmark in form of expected AQL-Answers. This offers the advantage, that the ground truth is automatically comparable to actual answers given by tools or cooperative analyses. Furthermore, a ground truth specified that way is more precise than the definitions that were frequently used in the past. For example, often it was only defined that a certain number of taint flows can be found but not which taint flows precisely. By comparing actual and expected AQL-Answers we can count true/false positive and negative findings. These countings can than be used to compute metrics such as precision, recall and F-measure, that allow us to competitively evaluate and compare (cooperative) analyses.

This concept of reproducible and automatic benchmarks was first presented in a reproducibility study [6]. Then used in the concept of cooperative analysis [7]. Some of the results determined there will be presented in the next section.

Evaluation. This benchmarking concept has been used to conduct a thorough evaluation of 13 standalone analysis tools as well as six cooperative analyses that employ in total 32 analysis tools, including, e.g., a newly developed static Android app slicer [8]. To do so, various benchmarks were used, imprecise ground truth definitions were refined and a whole new real-world malware benchmark, namely TaintBench [2] was developed. All details can be found in the thesis. In this summary, we want to provide a preview, therefore, we show that the standalone tool FlowDroid can be improved in various areas once it is combined with other tools in a cooperative analysis. In that sense, it is possible to add (1) advanced ICC as well as *inter-app communication (IAC)* capabilities, (2) the ability to analyze native code elements, and (3) to resolve reflection prior to analysis. Table 1 entitles the improvement with numbers for the DroidBench [1] micro benchmark.

Conclusion This summary exemplifies what is presented in-depth in the thesis: cooperative analyses are able to outperform standalone analysis tools precision-wise, and allow adding analysis capabilities with respect to certain analysis challenges. With reproducible and automatic benchmarks, these facts could be measured. In conclusion, cooperative analyses appear to be an adequate instrument to win the arms race between attackers and defenders.

Table 1: FlowDroid VS. Cooperative Analysis

DroidBench Category	FlowDroid[1]	Coop. Ana.	Δ
IAC	0.000	0.625	▲ 0.625
ICC	0.348	0.727	▲ 0.379
Native	0.000	0.889	▲ 0.889
Reflection	0.200	0.800	▲ 0.600

[1]: FlowDroid (2.9.0)

The thesis' artifact which includes all newly developed tools, results and further material (e.g., benchmarks and reproduction instructions) is publicly available at Zenodo [5].

References

[1] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, and P. D. McDaniel. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *Proceedings of PLDI, 2014*. ACM, 2014.

[2] L. Luo, F. Pauck, G. Piskachev, M. Benz, I. Pashchenko, M. Mory, E. Bodden, B. Hermann, and F. Massacci. Taintbench: Automatic real-world malware benchmarking of android taint analyses. *Empir. Softw. Eng.*, 27(1):16, 2022.

[3] F. Pauck. Aql-system (last accessed 27.04.2024). https://FoelliX.github.io/AQL-System.

[4] F. Pauck. *Cooperative Android App analysis.* PhD thesis, Paderborn University, Germany, 2023.

[5] F. Pauck. *Cooperative Android App Analysis (Thesis Artifact).* PhD thesis, Paderborn University, Jan. 2023.

[6] F. Pauck, E. Bodden, and H. Wehrheim. Do android taint analysis tools keep their promises? In *Proceedings of the 26th ESEC/FSE, 2018.* ACM, 2018.

[7] F. Pauck and H. Wehrheim. Together strong: cooperative android app analysis. In *Proceedings of ESEC/FSE, 2019.* ACM, 2019.

[8] F. Pauck and H. Wehrheim. Jicer: Simplifying cooperative android app analysis tasks. In *Proceedings of the 21st SCAM, 2021.* IEEE, 2021.