Jicer: Slicing Android Apps for Cooperative Analysis

Felix Pauck[1] Heike Wehrheim[2]

Abstract: Slicing allows to identify which program parts influence or are influenced by a certain statement of a program. Hence, if we know which statement is potentially causing an issue we can slice accordingly to only inspect the slice while debugging. With JICER, we proposed a slicer that can be used in a different context, namely cooperative Android app analysis. In combination with taint analysis tools, we employed JICER to get more accurate results.

Keywords: Cooperative Analysis; Android; Taint Analysis; Static Slicing

1 Cooperative Analysis with JICER

Android has become the most-used operating system, consequently it has also become a compelling target for attackers who, for example, try to steal users' private data. One instrument to detect privacy leaks before they are exploited is taint analysis. Fortunately, there exist many Android (taint) analysis tools that focus on different aspects related to the Android framework or an app's programming language. Even more luckily, there nowadays exist cooperative analysis frameworks that allow to compose (and evaluate) analysis combinations [PW19, PBW18]. With JICER [PW21] (Jimple Slicer) we proposed a static Android app slicer which is usable in cooperative analysis context. It allows slicing an app such that it can still be analyzed thereafter — a feature that distinguishes JICER form related Android slicing approaches. In the following the approach behind JICER is explained in more detail.

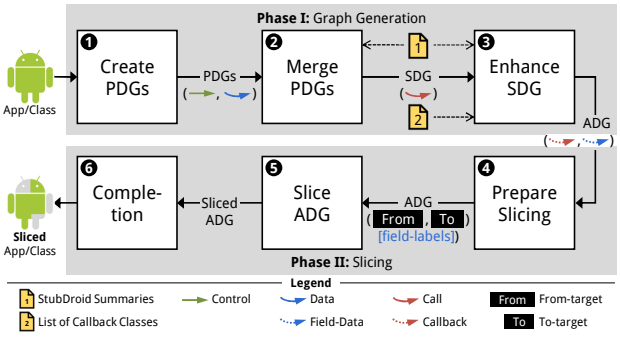

Fig. 1: Overview of the approach

JICER implements a six-step workflow that can be divided into two phases: graph generation and slicing (see Figure 1).

[1] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany fpauck@mail.uni-paderborn.de
[2] University of Oldenburg, Ammerländer Heerstraße 114–118, 26129 Oldenburg, Germany heike.wehrheim@uol.de

Once an Android app (`.apk` file) is provided as input, a *program dependence graph (PDG)*, modeling control and data dependencies, is computed for all methods contained in this app (❶). The second step (❷) merges these PDGs into a single *system dependence graph (SDG)* as proposed by Horwitz et al. back in 1990 [HRB90]. This SDG is then enhanced with edges that model callbacks (with respect to e.g. life-cycles or GUI elements) and data dependencies related to fields. The output of this enhancement step (❸) is called *app dependence graph (ADG)*. With the generation of the ADG the graph generation phase ends.

To start the slicing phase, the slicing criteria provided by the user are identified first (❹). They define *from* where and *to* where to slice. Note that JICER allows to provide a to-criterion, a from-criterion or both. Depending on the criteria given, a backward slice (to), a forward slice (from) or a chop (from-to) is computed. Figuratively speaking, a chop can be understood as the intersection of a backward and a forward slice. Once the ADG is sliced (❺), finishing touches are performed during Step ❻. Statements that are mandatory for analyses but not included in the slice are added to the slice. For example, `setContentView` statements link a layout to an Android activity, hence, many analyses require such statements to be available to know which layout must be taken into account. At the end, the sliced Android package (`.apk` file) is output. JICER also supports other output formats but `.apk` files are the preferred choice in cooperative analysis context, because most analysis tools require `.apk` files as input. Accordingly, all these tools can be used to analyze slices produced by JICER.

The evaluation presented in the proposing paper [PW21] shows that JICER is able to slice real-world apps thereby reducing app size about ~55 to ~96%. Most importantly, in a cooperative analysis JICER can increase the overall precision (eliminating up to 82% of false positives that have been found without slicing).

2 Data Availability

An artifact submitted along with the JICER study, that has successfully undergone an artifact evaluation, is available at Zenodo (`https://doi.org/10.5281/zenodo.5462859`). It contains all tools and results determined in the original study. Furthermore, the up-to-date version of JICER can be found at Github (`https://FoelliX.github.io/Jicer`).

Bibliography

[HRB90] Horwitz, Susan; Reps, Thomas W.; Binkley, David W.: Interprocedural Slicing Using Dependence Graphs. ACM Trans. Program. Lang. Syst., 12(1):26–60, 1990.

[PBW18] Pauck, Felix; Bodden, Eric; Wehrheim, Heike: Do Android taint analysis tools keep their promises? In: Proceedings of the 26th ESEC/FSE, 2018. ACM, 2018.

[PW19] Pauck, Felix; Wehrheim, Heike: Together strong: cooperative Android app analysis. In: Proceedings of ESEC/FSE, 2019. ACM, 2019.

[PW21] Pauck, Felix; Wehrheim, Heike: Jicer: Simplifying Cooperative Android App Analysis Tasks. In: Proceedings of the 21st SCAM, 2021. IEEE, 2021.