GenBenchDroid: Fuzzing Android Taint Analysis Benchmarks

Stefan Schott[1] Felix Pauck[2]

Abstract: The conventional approach of assessing the performance of Android taint analysis tools consists of applying the tool to already existing benchmarks and calculating its performance on the contained benchmark cases. Creating and maintaining a benchmark requires a lot of effort, since it needs to comprise various analysis challenges, and since each benchmark case needs a well documented ground-truth — otherwise one cannot know whether a tool's analysis is accurate. This effort is further increased by the frequently changing Android API. All these factors lead to the same, usually manually created, benchmarks being reused over and over again. In consequence analysis tools are often *over-adapted* to these benchmarks.

To overcome these issues we propose the concept of *benchmark fuzzing*, which allows the generation of previously unknown and unique benchmarks, alongside their ground-truths, at evaluation time. We implement this approach in our tool GenBenchDroid and additionally show that we are able to find analysis faults that remain uncovered when solely relying on the conventional benchmarking approach.

Keywords: Fuzzing; Benchmarks; Android Taint Analysis

1 Fuzzing Benchmarks with GenBenchDroid

Taint flows typically consist of a source, which introduces sensitive data to the app, a sink, which leaks sensitive data to the outside world and some intermediate data flow that connects source and sink. This data flow often comprises various programming *aspects*, like different data structures or multi-threading. The more complex the data flow is, the harder it is for analysis tools to uncover the taint flow. Thus, a proper benchmark needs to comprise benchmark cases with various degrees of complexity and analysis challenges (aspects). To be able to generate such Android apps, that can be used as benchmark cases, we split aspects that may possibly be contained inside taint flows into a set of *modules*. GenBenchDroid [SP22] interweaves these modules by inserting them into a *template*, which denotes the starting structure of an Android app. This allows GenBenchDroid to generate Android apps that comprise taint flows of various complexities.

Figure 1 shows an overview of GenBenchDroid's architecture. The *Fuzzer* component uses a grammar that knows about the aforementioned templates and modules (building blocks). This grammar is used to generate a *Benchmark Case Blueprint* (BCB),

[1] Paderborn University, Germany, Warburger Str. 100, 33098 Paderborn, Germany, stefan.schott@upb.de
[2] Paderborn University, Germany, Warburger Str. 100, 33098 Paderborn, Germany, fpauck@mail.upb.de

which specifies building blocks and their desired insertion order. The *Benchmark Case Generator* (BCG) component interweaves the building blocks that are specified in the BCB and generates an Android app, as well as the corresponding ground-truth. This ground-truth is determined by generating a graph that represents the used building blocks and by finding paths in this graph that connect source and sink modules.

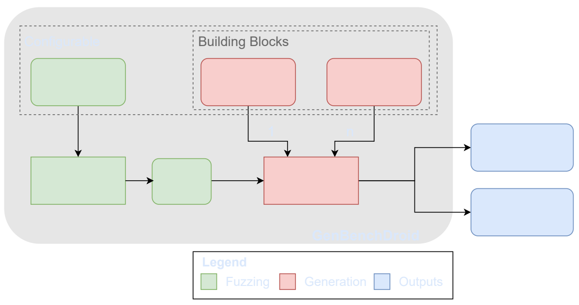

Fig. 1: Overview of GenBenchDroid

Benchmark Fuzzing offers many advantages that can improve and complement conventional benchmarking approaches. By design, benchmark fuzzing decreases the likelihood of over-adaptation, as benchmark cases are not known before evaluation time. Furthermore, our experiments on state-of-the-art taint analysis tools (FlowDroid [Ar14] and Amandroid [WRO18]) uncovered previously unknown analysis defects. We were able to uncover a scalability issue in Amandroid by generating benchmark cases of various sizes. Additionally, we uncovered analysis defects in FlowDroid and Amandroid that would only show up, if aspects appear in combination inside a single taint flow. These defects can hardly be uncovered by only relying on conventional benchmarks, as it is impossible to manually create arbitrary aspect combinations and since these aspects used in isolation are analyzed correctly by both tools.

2 Data Availability

GenBenchDroid and its source code, as well as all data that is related to our performed experiments is available at `https://doi.org/10.5281/zenodo.7023084`. An up-to-date version of GenBenchDroid can be found on Github (`https://github.com/stschott/GenBenchDroid`).

Bibliography

[Ar14] Arzt, Steven; Rasthofer, Siegfried; Fritz, Christian; Bodden, Eric; Bartel, Alexandre; Klein, Jacques; Le Traon, Yves; Octeau, Damien; McDaniel, Patrick: Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. Acm Sigplan Notices, 49(6):259–269, 2014.

[SP22] Schott, Stefan; Pauck, Felix: Benchmark Fuzzing for Android Taint Analyses. In: 22nd IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2022, Limassol, Cyprus, October 3-4, 2022. IEEE, 2022. To appear.

[WRO18] Wei, Fengguo; Roy, Sankardas; Ou, Xinming: Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps. ACM Transactions on Privacy and Security (TOPS), 21(3):1–32, 2018.