Anne Koziolek, Ina Schaefer, Christoph Seidl (Hrsg.): Software Engineering 2021,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 23

Heaps'n Leaks: How Heap Snapshots Improve Android Taint Analysis

Manuel Benz[1], Erik Krogh Kristensen[2], Linghui Luo[3], Nataniel P. Borges Jr.[4], Eric Bodden[5], Andreas Zeller[6]

Abstract: The assessment of information flows is an essential part of analyzing Android apps, and is frequently supported by static taint analysis. Its precision, however, can suffer from the analysis not being able to precisely determine what elements a pointer can (and cannot) point to. Recent advances in static analysis suggest that incorporating dynamic heap snapshots, taken at one point at runtime, can significantly improve general static analysis. In this paper, we investigate to what extent this also holds for *taint* analysis, and how various design decisions, such as when and how many snapshots are collected during execution, and how exactly they are used, impact soundness and precision. We have extended FlowDroid to incorporate heap snapshots, yielding our prototype Heapster, and evaluated it on DroidMacroBench, a novel benchmark comprising real-world Android apps that we also make available as an artifact. The results show 1. the use of heap snapshots lowers analysis time and memory consumption while increasing precision; 2. a very good trade-off between precision and recall is achieved by a *mixed mode* in which the analysis falls back to static points-to relations for objects for which no dynamic data was recorded; and 3. while a *single* heap snapshot (ideally taken at the end of the execution) suffices to improve performance and precision, a better trade-off can be obtained by using multiple snapshots.

Keywords: points-to analysis; heap snapshot; taint analysis; Soot; Android

1 Introduction

Android is the world's most popular mobile operating system. Its official marketplace, Google Play Store, holds more than 3.3 million apps, which can be installed on billions of devices. To perform their tasks, apps frequently interact with sensitive information—from private images to banking details. Research shows that security-related bugs introduced by developers frequently put this sensitive information at risk [En11; En14; Gr12; Ra15].

To identify such sensitive information leaks, *taint analysis* detects potential leaks by determining if data acquired on a sensitive source reaches a sink, where the information would no longer be secure. Such taint flows can be detected statically or dynamically. A *static* taint analysis, which we focus on in this paper, reasons about all possible execution

[1] Paderborn University, Department of Computer Science manuel.benz@codeshield.de
[2] Aarhus University, Department of Computer Science erik@cs.au.dk
[3] Paderborn University, Department of Computer Science linghui.luo@upb.de
[4] CISPA Helmholtz Center for Information Security, Department of Computer Science nataniel.borges@cispa.saarland
[5] Paderborn University & Fraunhofer IEM, Department of Computer Science eric.bodden@upb.de
[6] CISPA Helmholtz Center for Information Security, Department of Computer Science zeller@cispa.saarland

doi:10.18420/SE2021_02

paths in a program and aims to achieve (close to) perfect recall, i.e., it seeks to identify virtually all potentially sensitive information leaks. Static analyses, though, often suffer from a trade-off between accuracy and scalability. Although existing taint analysis tools such as FlowDroid [Ar14] can be configured to conduct a relatively precise flow, context, and field-sensitive analysis, such configuration needs to be identified by possibly inexperienced users—and imprecise configuration causes the taint analysis to report substantial amounts of false positives [LBS18].

A recent approach by Grech et al. addresses this problem by extending static pointer analysis with information extracted from *heap snapshots*, collected at runtime. As the authors show, one can improve soundness [Gr17] by augmenting statically computed points-to information with *additional* data from the heap snapshots. Conversely, one can improve precision by *restricting* static points-to computation to such information present in the heap snapshots [Gr18].

In this work, we present an empirical study in which we seek to reproduce the original experiments revised by Grech et al. but also go significantly beyond them to address these open questions. We make the following original contributions:

Using heap snapshots for Android taint analysis. We investigate how heap snapshots impacts the soundness and precision, not just of simple pointer analysis, but of a concrete client analysis, a *static Android taint analysis.*

Assessment of design decisions. We investigate how various essential design decisions impact precision and soundness of the analysis. In particular, we evaluate the impact of two novel extensions:

- information not only from a *single* heap snapshot but *multiple ones,* e.g., collected at various times during the execution; and
- *dynamic* heap models collected at runtime (precise, but possibly unsound) versus pure *static* heap models (sound, but possibly imprecise) versus *mixed* models that seek to define a sensible middle ground between those two extremes by focusing on precision and enhancing a dynamic model with static information.

Implementation and Benchmark. To evaluate the above decisions, we implemented *Heapster*, an extension to FlowDroid that can incorporate heap dumps. Additionally, we created *DroidMacroBench*, a set of 12 real-world Android applications that we manually labeled with ground truth for taint analyses.

Evaluation. We explore the impact of different design decisions about *when to collect and how to consume heap snapshots*. In our evaluation we show that:

- adding heap snapshots can significantly improve the precision of taint analysis (from 50.3% to up to 94.7%);
- while restricting points-to information to that of the heap snapshots offers high precision it significantly harms recall. Our mixed mode solution, however, provides

both good precision (77.1%) and good recall (68.4%). Its F1 score is the highest among all configurations;

- in all evaluated scenarios, incorporating heap snapshots significantly lowers the amount of computational resources required by the taint analysis, moreover, in 90% of the scenarios it also improves the analysis performance; and
- while a single heap snapshot, taken at the end of the runtime, suffices to significantly increase the analysis precision, additional snapshots, taken at different times, are beneficial for the analysis recall, achieving the best overall F1 score.

For details please consider the full paper accessible at `https://dl.acm.org/doi/10.1145/3377811.3380438` and `https://www.hni.uni-paderborn.de/pub/10027`.

References

[Ar14] Arzt, S.; Rasthofer, S.; Fritz, C.; Bodden, E.; Bartel, A.; Klein, J.; Traon, Y. L.; Octeau, D.; McDaniel, P. D.: FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In: ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014. Pp. 259–269, 2014, URL: `http://doi.acm.org/10.1145/2594291.2594299`.

[En11] Enck, W.; Octeau, D.; McDaniel, P. D.; Chaudhuri, S.: A Study of Android Application Security. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings. USENIX Association, 2011, URL: `http://static.usenix.org/events/sec11/tech/full%5C_papers/Enck.pdf`.

[En14] Enck, W.; Gilbert, P.; Chun, B.-G.; Cox, L. P.; Jung, J.; McDaniel, P.; Sheth, A. N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones William. Communications of the ACM 57/3, pp. 99–106, 2014, ISSN: 00010782, arXiv: `1005.3014`, URL: `http://dl.acm.org/citation.cfm?doid=2566590.2494522`.

[Gr12] Grace, M. C.; Zhou, W.; Jiang, X.; Sadeghi, A.: Unsafe exposure analysis of mobile in-app advertisements. In (Krunz, M.; Lazos, L.; Pietro, R. D.; Trappe, W., eds.): Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC 2012, Tucson, AZ, USA, April 16-18, 2012. ACM, pp. 101–112, 2012, URL: `https://doi.org/10.1145/2185448.2185464`.

[Gr17] Grech, N.; Fourtounis, G.; Francalanza, A.; Smaragdakis, Y.: Heaps don't lie: countering unsoundness with heap snapshots. PACMPL 1/OOPSLA, 68:1–68:27, 2017, URL: `https://doi.org/10.1145/3133892`.

[Gr18] Grech, N.; Fourtounis, G.; Francalanza, A.; Smaragdakis, Y.: Shooting from the heap: ultra-scalable static analysis with heap snapshots. Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2018/, pp. 198–208, 2018, URL: http://dl.acm.org/citation.cfm?doid=3213846.3213860.

[LBS18] Luo, L.; Bodden, E.; Späth, J.: A Qualitative Analysis of Taint-Analysis Results, tech. rep., Heinz Nixdorf Institute, Paderborn University, Aug. 2018.

[Ra15] Rasthofer, S.; Arzt, S.; Hahn, R.; Kohlhagen, M.; Bodden, E.: (In)Security of Backend-as-a-Service. In: blackhat europe 2015. Nov. 2015, URL: http://bodden.de/pubs/rah+15backend.pdf.