Journal of Computer Languages 52 (2019) 25

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.editorialmanager.com/cola/default.aspx

Preface

Static analysis of android apps: Security and privacy

When installing an app on an Android device, we grant it access to parts of our personal (and often sensitive) data that are either stored on the device or dynamically acquired by the environment (e.g., GPS coordinates). How can we detect whether such information is leaked by the app to (unauthorized) external parties? How can we guarantee the absence of security and privacy threats when downloading an app from the store?

The aim of this special issue is to provide a picture of how static analysis techniques address the abovementioned challenges: tracing how information is potentially released; how problematic such releases are (as a function of the declassification, or anonymization, applied to the data); whether and how side channels (i.e., implicit dependencies) are a threat; and how to prevent undesirable interactions due to multiple applications running on the same device or on connected devices.

The special issue consists of two papers, selected out of the manuscripts submitted in response to the call for papers.

The first contribution, titled "A Scalable, Flow-and-Context-Sensitive Taint Analysis of Android Applications" by Wontae Choi, Jayanthkumar Kannan, and Domagoj Babic presents a novel static taint analysis that performs context-sensitive, flow-sensitive, and multi-source tracking. The most critical challenge with such a precise analysis is to preserve the overall scalability of the tool. Therefore, the approach has mostly been tailored towards various optimizations. Ad-hoc symbolic summaries are applied to obtain precise inter-procedural analysis, while the intra-procedural analysis has been optimized in various ways (including a pruning technique that combines escape analysis, a framing technique, and a novel form of bypassing certain code). Overall the combination of these techniques allows to run this analysis on the bulk of existing Android applications in a reasonable amount of time. The paper further presents a deep experimental evaluation, focusing on the analysis' execution time.

The second paper, titled "Jitana: A Modern Hybrid Program Analysis Framework for Android Platforms", by Yutaka Tsutano, Shakthi Bachala, Witawas Srisa-an, Gregg Rothermel, and Jackson Dinh, introduces a hybrid analysis framework for Android applications. Jitana is designed for collections of applications that may interact with one another. It preserves analysis graphs such as CFG, DFG, etc., so these can be used when (co-)analyzing the set of applications. This is unlike most existing analyses, which compute these on a per-application basis to find entry points and exit points, but do not persist them for actual inter-application analysis. Use of hybrid analysis enables the approach to handle dynamically-loaded code as well as to compute and display code coverage during dynamic analysis. In particular, a visualization plugin, TraVis, allows coverage to be displayed in real-time as an application is being executed.

We are indebted to colleagues that helped us in the reviewing process and to the Editor in Chief, Prof. Marjan Mernik, for his strong and constant support.

Guest Editors

Agostino Cortesi[a], Omer Tripp[b]
[a] *Ca' Foscari University, Venice, Italy*
[b] *Google Inc, Amazon, USA*
E-mail addresses: cortesi@unive.it (A. Cortesi),
trippo@google.com (O. Tripp).

https://doi.org/10.1016/j.cola.2019.03.005

Available online 17 April 2019