Software Engineering Seminar

Symbolic Side-Channel Analysis

Description

Symbolic Execution is a program analysis, which uses symbolic values (instead of concrete ones) to execute the program under test. The symbolic analysis leads to a systematic exploration of the search space, which can be directed by various heuristics. Recent work like *CoCo-Channel* by Brennan et al. [1] and Malacaria et al. [4] leverages symbolic execution for the detection of side-channel vulnerabilities. For example, CoCo-Channel [1] provides a compositional generation of symbolic path cost expressions, which is used to find imbalanced program paths. *Adversarial Symbolic Execution* by Guo et al. [3] focuses specifically on concurrency-related cache timing leaks. The *CacheFix* framework [2] makes a step further by also attempting to fix a potential side-channel vulnerability: It either verifies the cache side-channel freedom of the program or synthesizes a series of patches to ensure cache side-channel freedom during program execution.

The goal of this seminar topic is to collect the current research directions in symbolic side-channel analysis. Therefore, it is is necessary to perform an initial literature analysis based on the provided publications. The student should examine and discuss the approaches given in the papers and compare them to each other and to similar existing techniques. Additionally, the student is asked to provide a critical discussion of the current research directions, which should also include an outlook for possible future work.

References

[1] Tegan Brennan, Seemanta Saha, Tevfik Bultan, and Corina S. Păsăreanu. Symbolic path cost analysis for side-channel detection. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2018, pages 27–37, New York, NY, USA, 2018. ACM.

[2] S. Chattopadhyay and A. Roychoudhury. Symbolic verification of cache side-channel freedom. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2812–2823, Nov 2018.

[3] Shengjian Guo, Meng Wu, and Chao Wang. Adversarial symbolic execution for detecting concurrency-related cache timing leaks. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, pages 377–388, New York, NY, USA, 2018. ACM.

[4] P. Malacaria, M. Khouzani, C. S. Pasareanu, Q. Phan, and K. Luckow. Symbolic side-channel analysis for probabilistic programs. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 313–327, July 2018.

Contacts

Yannic Noller (`noller@informatik.hu-berlin.de`)
Software Engineering Group
Institut für Informatik
Humboldt-Universität zu Berlin