Bachelor/Master Thesis Topic

SymCC for Java: Symbolic execution via Java bytecode instrumentation

Motivation and Background:

Symbolic execution is a powerful tool for generating test cases and detecting software bugs. In Java, the current state-of-the-art symbolic execution tool is Symbolic PathFinder (SPF) [ASE’2010]. SPF essentially performs a custom interpretation of Java bytecode to compute the symbolic expressions and path constraints. However, it is also known that symbolic execution via interpretation is very expensive. Recent work introduced SymCC [USENIXSec’2020], a symbolic execution engine that instruments the program to include the symbolic execution computation in the binary instead of interpreting it. It was shown that such an approach could lead to dramatic performance gains. This project aims to investigate and develop a similar strategy for the symbolic execution of Java bytecode.

Student Task and Responsibilities:

- Java tool for the symbolic execution of Java bytecode via instrumentation.
- Note that it is not required to implement a full-fledged symbolic executor. This project aims at providing the initial infrastructure and is supposed to implement a working prototype for a specific usage scenario, which can be agreed upon.
- Documented findings of the conducted experiments.

Deliverables:

- Evaluation artifacts (dataset, testing tools, drivers, etc.)
- Documented findings of the conducted experiments. Of particular interest are: bugs in sanitizers, inconsistencies between reported documentation and actual behavior, identified usability issues, and recommendations for their usage.

Pre-Requisites: (Programming Languages, OS, Skills, Papers, etc)

Knowledge of Java and Java bytecode instrumentation is helpful for this project. Further, it would be beneficial if the student had some experience with symbolic execution.

[ASE’2010] Corina S. Păsăreanu and Neha Rungta. 2010. Symbolic PathFinder: symbolic execution of Java bytecode. In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE '10).

[USENIXSec’2020] Poeplau, S. and Francillon, A., 2020. Symbolic execution with SymCC: Don't interpret, compile!. In 29th USENIX Security Symposium (USENIX Security 20) (pp. 181-198).
[SP’2019] D. Song *et al.*, "SoK: Sanitizing for Security," *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2019, pp. 1275-1295, doi: 10.1109/SP.2019.00010.

[Software’2021] M. Boehme, C. Cadar and A. Roychoudhury, "Fuzzing: Challenges and Reflections" in IEEE Software, vol. 38, no. 03, pp. 79-86, 2021.

Contacts

Prof. Dr. Yannic Noller (`sq-office@rub.de`)

Software Quality group, Faculty of Computer Science, Ruhr University of Bochum