Applying Heuristics to Improve our Java Symbolic Execution Engine

Edit Pengő[a]

[a] Department of Software Engineering, University of Szeged
pengoe@inf.u-szeged.hu

Abstract

The common approach to detect runtime errors in software systems is testing. However, testing can be expensive and might not cover every states of a system. The well-known symbolic execution technique provides one solution for this problem. By executing a program symbolically we traverse in a symbolic execution tree consisted of all possible execution paths. Along the branches we can find the runtime errors with the exact call stack leading to them.

We developed a symbolic execution engine, called RTEHunter, for detecting runtime failures in Java programs. It is a static code analyser tool so it has many limitations due to the nature of symbolic execution. For example, in practice it is impossible to explore the whole symbolic execution tree therefore constraints must be applied (e.g. limiting the traversing depth in the tree or the execution time). This means that although many real problems can be discovered in the source code, in some cases it gives false positive hits as well.

Our goal is to improve the results of RTEHunter by eliminating certain kinds of false positive hits. We describe different approaches for this and to prove its practical benefits the results are validated on more than 200 various sized Java systems.

Keywords: Source code analysis, Symbolic execution, Java

MSC: 68N01