Aiman Priester

KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs (P03)

Summary

- The paper discusses an execution tool that generates tests for high level programs. The main idea is to get most, if not all, of the edge cases in larger, more demanding programs at runtime. Programs with command line arguments are especially tested. The paper suggests that program line programs that invoke memory locations are likely to cause security issues when referencing or dereferencing variables. While it is trivial to have these tests on a small scale, KLEE aims to improve scalability to real world code. The paper ends by explaining the results of bug finding in BUSYBOX and MINX tools, which was fairly substantial.

Strengths

- Since KLEE essentially does state saving and testing, concurrent states can be executed at the same time, allowing for faster and less time consuming testing scaled to the performance of the testbench computer.
- KLEE coexists with a currently running filesystem. It does not intrude in disallowed memory locations. This is an advantage compared to other coverage tests compared in the paper.

Weaknesses

- KLEE is likely to break in test suites without root access. While it is understandable that one would not want to allow root access for a program like this, however, it leaves the question out in the open about the variability of this in already heavily developed programs. It is insinuated that KLEE would probably not work on legacy programs.

Unresolved Issues

- KLEE does not allow for dynamically allocated memory objects as discussed in section 3. Most of the efficient programs out in the market place are heavily reliant on the ability to dynamically allocate and reallocate memory as necessary.

Discussion

- Reporting is done based on line coverage, which is probably not the best / most efficient way of reporting. What could be used in replacement of this method?