Fine-Grained Coverage-Based Fuzzing

Wei-Cheng Wu[1,2], Bernard Nongpoh[1], Marwan Nour[1],
Michaël Marcozzi[1], Sébastien Bardin[1], and Christophe Hauser[2]

[1] Université Paris-Saclay, CEA, List, France
[2] University of Southern California, USA

Extented abstract. Fuzzing is a popular software testing method that discovers bugs by massively feeding target applications with automatically generated inputs. Many state-of-the-art fuzzers use branch coverage as a feedback metric to guide the fuzzing process. The fuzzer finds and retains inputs for further mutation only if branch coverage is increased. However, branch coverage only provides a shallow sampling of program behaviours and hence may discard interesting inputs to find and mutate.

This work aims at taking advantage of the large body of research over defining finer-grained code coverage metrics (such as control-flow, data-flow or mutation coverage) and at evaluating how fuzzing performance is impacted when also using these metrics to find and select interesting inputs for mutation. We propose to make branch coverage-based fuzzers support most fine-grained coverage metrics out of the box (i.e., without changing fuzzer internals). We achieve this by making the test objectives defined by these metrics (such as conditions to activate or mutants to kill) explicit as new branches in the target program. Fuzzing such a modified target is then equivalent to fuzzing the original target, but the fuzzer will also retain inputs covering the additional metrics objectives for mutation. In addition, all the fuzzer mechanisms to penetrate hard-to-cover branches will help covering the additional fine-grained objectives.

We use the proposed approach to evaluate the impact of supporting two fine-grained coverage metrics (multiple condition coverage and weak mutation) over the performance of two state-of-the-art fuzzers (AFL++ and QSYM) with the standard LAVA-M and MAGMA benchmarks, totalling more than seven hundreds of thousands of lines of code, for two and a half years of CPU time.

Our exeperiments suggest that our mechanism for runtime fuzzer guidance, where the fuzzed code is instrumented with additional branches, is effective. More efforts are thus needed to make it even better integrated with fuzzer instrumentation and to use it to encode guidance from human users or static analysers. Our results also show that the impact of fine-grained metrics over fuzzing performance is hard to predict before fuzzing, and most of the time either neutral or negative. As a consequence, we do not recommend using them to guide fuzzers in general. Yet, our work also calls for further research to better investigate situations where favourable circumstances would make fine-grained metrics profitable for fuzzing, like when only fragile or sensitive parts of the codebase would be instrumented, or as a complement to classical fuzzing campaigns.

Presenters. This paper would be presented by **Wei-Cheng Wu** and **Michaël Marcozzi**.

Pointer to the original paper. The orignal paper can be found on the ACM digital library at `https://dl.acm.org/doi/10.1145/3587158`.

Journal-first classification. This paper was published within the ACM TOSEM Registered Papers track, where papers must qualify as journal-first papers as a submission requirement (see `https://dl.acm.org/journal/tosem/registered-papers`).