Driller: Augmenting Fuzzing Through Selective Symbolic Execution

Nick Stephens, John Grosen, Christopher Salls, Audrey Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna

Writer: Kai Li

1 Summary

This paper proposed a augmented fuzzing tool **Driller** [1] which complements fuzzing technique with concolic execution to enable the fuzzers to pass complex checks in binaries, thus guides the fuzzers to explore deeper and identify vulnerabilities that using traditional fuzzers is hard to discover. Specifically, it targets the binaries that require both *generic inputs* and *specific inputs*(exact match), fuzzing works well to explore *generic input* while concolic execution works well to solve the *specific input*, therefore, combining concolic execution with fuzzing can explore deeper in the program and achieve more code coverage (more crashes) than traditional fuzzing techniques. To solve the path explosion problem when applying concolic execution, it proposes selective concolic execution by exploring based on the input provided by the fuzzer. Driller is implemented based on AFL and angr and evaluated by the DARPA Cyber Grand Challenge, the results show that Driller can find 8.4% more crashes than the union of baseline techniques (fuzzing alone and symbolic execution alone).

2 Strengths and Weaknesses

2.1 Strengths

1. An effective technique to improve the fuzzing by leveraging selective concolic execution to reach deeper in the program, and improve the scalability of concolic execution by using fuzzing to solve path explosion.
2. The Driller tool is implemented and can be evaluated on the real-world binaries.
3. Strong evaluation result shows the effectiveness of Driller that it can achieve more code coverages and find more crashes than the baseline techniques, and winning the PARDA CGC.

2.2 Weakness

1. A major limitation of this tool is the symbolic explorer stub that used to reduce the number of expensive concolic engine invocations, which makes Driller to miss some paths in some cases.
2. In a case when user input is treated as generic input in one component and specific input in another, this can make Driller become in-effetive as the fuzzing is disabled.
3. Driller scores the same with the first-place in the DARPA challenge while Driller is implemented and tested in a more relaxed circumstance.

References

[1] Nick Stephens, John Grosen, Christopher Salls, Andrew Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Driller: Augmenting fuzzing through selective symbolic execution. In *NDSS*, volume 16, pages 1–16, 2016.