Artifact for "CONFETTI: Amplifying Concolic Guidance for Fuzzers"

Abstract of paper

Fuzz testing (fuzzing) allows developers to detect bugs and vulnerabilities in code by automatically generating defect-revealing inputs. Most fuzzers operate by generating inputs for applications and mutating the bytes of those inputs, guiding the fuzzing process with branch coverage feedback via instrumentation. Whitebox guidance (e.g., taint tracking or concolic execution) is sometimes integrated with coverage-guided fuzzing to help cover tricky-to-reach branches that are guarded by complex conditions (so-called "magic values"). This integration typically takes the form of a targeted input mutation, for example placing particular byte values at a specific offset of some input in order to cover a branch. However, these dynamic analysis techniques are not perfect in practice, which can result in the loss of important relationships between input bytes and branch predicates, thus reducing the effective power of the technique.

CONFETTI introduces a new, surprisingly simple, but effective technique, *global hinting*, which allows the fuzzer to insert these interesting bytes not only at a targeted position, but in any position of any input. We implemented this idea in Java, creating CONFETTI, which uses both targeted and global hints for fuzzing. In an empirical comparison with two baseline approaches, a state-of-the-art greybox Java fuzzer and a version of CONFETTI without global hinting, we found that CONFETTI covers more branches and finds 15 previously unreported bugs, including 9 that neither baseline could find.

Purpose of the research artifact

We created this artifact to allow future researchers to:

1. evaluate their fuzzer in comparison to ours by running our fuzzer
2. check that the output that they obtain from repeating our experiments matches our results
3. make modifications to and recompile our software in order to create a new fuzzer

Badges claimed

- Available
- Reusable
- Functional (it is sometimes unclear if Reusable implies Functional, and whether it is possible to obtain both badges or only one)

Technology skills assumed

Ideally, a reviewer of this artifact:

- Is comfortable working with a virtual machine, and running commands in a shell
- Is familiar with Java and Maven

Technological environment assumed

To use our virtual machine artifact, a reviewer *must*:

- Have at least 32 GB RAM free to assign to the VM
- Have at least 4 CPU cores to assign to the VM
- Have at least 100GB of storage to store the VM; more space may also be needed temporarily to extract the VM

If the reviewer would also like to build our software outside of the VM, then they should have:

- Mac OS X (11.x or 12.x) or Ubuntu (18 or 20) operating system
- Java 8 JDK installed