TEST CASE GENERATION USING FUZZING IN C++

ZOLTAN PORKOLAB
IA MGVDLIASHVILI
ELTE

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

How do we verify software?

Unit Testing

Advantages

1. Essential for writing quality software
2. Most straightforward to reason about compared to other methods
3. Faster to implement and run

Disadvantages

1. Not trivial to choose meaningful combinations
2. Hard to determine edge cases
3. Relies too much on the developer

Static & Dynamic Analysis

- Used for finding vulnerabilities
- Concentrates on edge cases, not on general logic of the application
- As opposed to unit testing, it is independent from the developer

Fuzzing

How fuzzing works

- gives the program some input (array of bytes)
- measures coverage that the bytes triggerred
- saves successful inputs and mutates them to generate new ones

What is accomplished

Exposes Interface vulnerabilities

- Ones that are hard to detect manually
- Achieves powerful results in very short time

In order to produce input, it uses

- Generational and mutation-based methods
- Coverage-guided engine

Coverage based fuzzing for generating test cases

Motivation

- Unit tests verify that functions behave as expected for a particular internal state
- If data encapsulation is used, internal state is the result of other function calls
- Therefore, we could treat a some test cases as a sequence of function calls on the member

What is different from fuzzing

Instead of using sequence of bytes as input, we will generate sequence of member function calls

Overall description of the solved problems

We need something that will

1. Generate sequences of calls (as strings)
2. Map the strings to function calls
3. Observe coverage resulting from these calls
4. Compare the results and determine ones that are worth saving

Demo

Summary

This program could enable developers to

- automatically generate minimal test cases with high coverage for their libraries
- have a generic way of testing the logic of the application, just like static and dynamic analysis for vulnerabilities

References

- Source code and this pdf.
  https://github.com/iarigby/bsc-thesis
- LLVM libFuzzer – a library for coverage-guided fuzz testing.
  https://llvm.org/docs/LibFuzzer
- Miller, Ch., Peterson, Z.N.J. Analysis of Mutation and Generation-Based Fuzzing. Whitepaper. 2007
  https://defcon.org/dc-15-miller-WP.pdf
- Hanno Böck, How Heartbleed could've been found, 2015
  https://blog.hboeck.de/868-How-Heartbleed-found

THANK YOU FOR ATTENTION

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE