Technische Universität München

Hybrid Greybox Fuzz Testing

Hiwi project proposal

Introduction

To automatically find vulnerabilities, we mainly consider two techniques –

1. Symbolic execution
    a. A deterministic method of enumerating program path (control flow graph)
    b. Program paths are represented as logical conditions (called *path constraints*)
    c. Solving path constraints leads to generation of test cases
2. Fuzzing
    a. Random testing but a bit smarter
    b. Starts with a few (manually generated) inputs (called *seed inputs*)
    c. Monitors system environment for triggers
    d. Mutates seed inputs to generate new triggers

Symbolic execution, due to its access to underlying code can be called a *whitebox* testing technique, whereas fuzzing, due to its blindness to program internals, can be called a *blackbox* technique.

However, current symbolic execution and fuzzing implementations have many problems, such as

1. Path explosion – When a program has too many paths for symbolic execution to explore, e.g. due to input dependant loops,
2. When the path constraints are too big for constraint solvers,
3. Manual generation of seed input consumes too much time, and
4. Input mutation fails to cover new paths in programs.

We propose a hybrid greybox technique to deal with all these problems and make automated testing faster and more efficient.

Your tasks

Difficulty mode: Easy

1. Use symbolic execution to generating seed inputs for fuzzing.

Difficulty mode: Moderate

1. Determine (empirically) “path constraint size vs. time” trend in symbolic execution.
2. Implement *constraint thresholding*.

Difficulty mode: Hard

1. *Flipper hybrid* strategy -
    a. Fuzzing when symbolic execution gets stuck
    b. Symbolic execution when fuzzing gets stuck
    c. ...

What we offer

1. Ample time to ramp up on and get familiar with involved technologies.
2. Easy access to existing tools and academic links/sources.
3. Weekly or bi-weekly meeting with feedback from your supervisor.
4. Flexible working hours.
5. If needed, desk and a quiet working space.
6. Possibility of publishing papers in top-tier conferences.

Desired qualification

1. Prior programming experience in Python and C/C++.
2. Interest in software testing.

If interested, please send your CV and transcripts of grades to ognawala@in.tum.de.