Lab 2: Experimenting with Symbolic Execution and Fuzzing

Objective

The objective of this lab is to experiment with symbolic execution and fuzzing tools on programs with inserted vulnerabilities and observe how well they can discover these vulnerabilities.

Test Programs

There are some small test programs at https://drive.google.com/drive/folders/1eJjatr_NXLuT7_LA7uwmIxTEVaS6dn66?usp=drive_link

Tools

You are expected to experiment with the following tools:

Klee: https://github.com/klee/klee

AFL++: https://github.com/AFLplusplus/AFLplusplus

Read the documentation for each tool. For Klee, you can directly use its container image from docker hub: https://hub.docker.com/r/klee/klee. For AFL++, you can also directly use its docker container.

Task 1: Klee (40%)

Use Klee on the test programs, answer the following questions: 1) Can Klee find inputs that pass all the checks in the programs? and 2) What inputs are generated in what order and why?

Task 2: AFL++ (30%)

Use the default configurations (one for source code based and one for binary) of AFL++ and answer the following questions: 1) Can it find inputs that pass all the checks in the programs? 2) What inputs are generated in what order and why? 3) Is there a significant difference between the source-based fuzzing and binary-based fuzzing? If so, why?

Task 3: AFL++ with SymQEMU as a custom mutator (30%)

Try SymQEMU as a custom mutator on the same test programs, and answer: 1) Can it find inputs that pass all the checks in the programs? 2) What inputs are generated in what order and why?3) Does it perform better than Klee and AFL++, and why?

A few important points:

1. The report should include figures and screenshots for important steps and results.
2. You are not expected to run each experiment for very long. Each experiment can be as short as five to ten minutes.
3. It is important to explain your observation. Simply reporting the results is not good enough.