CS 6501 Software Analysis and Applications

Project 1: Comparing AFL and Klee

Out: March 5, due March 26, 2020

(20 points/30 with extra credit)

Learning Objectives

1. Strengthen the understandings of fuzzing and symbolic execution
2. Get hands-on experience with the-state-of-the-art tools
3. Improve problem solving skills on testing
4. Work with real-world software and bugs

Description

In this homework, you are going to run and further study the testing techniques learned in our class. American Fuzzy Lop (AFL) is a fuzzing tool that has found many bugs in real-world software. KLEE is a symbolic execution tool that can automatically generate test inputs for covering as many branches as possible. We will use a few buggy programs to test and compare the performance of the two tools. Example codes and instructions are provided in 'softwareAnalysisProject_AFL_KLEE.zip' to help you get started. In the following, please find a list of steps to follow:

1. Install American Fuzzy Lop (AFL): http://lcamtuf.coredump.cx/afl/. Follow the instructions in 'softwareAnalysisProject_AFL_KLEE/AFL/AFL_instructions.md' to install AFL and learn how to use it.
   a) Download and install AFL
   b) Test the example program "test-instr.c"
2. Install KLEE: https://klee.github.io/. Follow the instructions in 'softwareAnalysisProject_AFL_KLEE/KLEE/KLEE_instructions.md' to install KLEE and learn how to use it.
   a) Easiest way to install KLEE is using the docker image: https://klee.github.io/docker/
   b) Go to klee_src/examples and run get_sign.c example following the tutorial in https://klee.github.io/tutorials/testing-function/
3. Compare AFL and KLEE on the get_sign example
   a) Since the test harness for KLEE and AFL are different, for testing with AFL, you need to modify the main function of the code provided by KLEE a little bit, you can use the file in 'softwareAnalysisProject_AFL_KLEE/AFL/get_sign.c'.
   b) Introduce a bug to get_sign.c and test the buggy version of get_sign.c with AFL and KLEE.
4. Compare AFL and KLEE on regexp.c example provided by klee: https://klee.github.io/tutorials/testing-regex/
   a) For testing with AFL, you can use

'softwareAnalysisProject_AFL_KLEE/AFL/ get_sign.c'. For testing with KLEE, the example program can be found in klee_src/examples in the docker container.

b) First, make sure the program itself is correct, e.g. compile it and try some inputs to see whether it gives you the correct regular expression matching result. Correct the program if needed.

c) Be careful with the problem with the default test harness in the file provided by KLEE (Read the last section in https://klee.github.io/tutorials/testing-regex/). Correct it accordingly. For AFL, this problem has been fixed in the provided file 'softwareAnalysisProject_AFL_KLEE/AFL/get_sign.c'.

d) Introduce two buggy versions of regexp.c by implanting two bugs, test regexp.c with both AFL and KLEE.

5. (Extra credit 10 points) Compare AFL and KLEE on a real-world program you want to test. It can be created by you or find from the open source repositories (tip: since KLEE is hard to set up, consider starting with the software that works with Klee, e.g., from coreutils.)
6. Write-up your studies – what happened and what did you learn.

Deliverables (20 pt./10 pt. with extra credit)

Please zip the following files and submit the zipped file to Collab.
From Step 3, you'll submit:

1. (2 pt) screenshots to show that get_sign.c ran successfully with KLEE and afl
2. (1 pt) a buggy version of get_sign.c and a readme file that explains where is the bug and what is the bug
3. (2 pt) a folder that contains the test inputs generated from AFL and klee
4. (2 pt) a folder that stores the output of running these test inputs on AFL and klee

From Step 4, you'll submit:

1. (2 pt) a correct version of regexp.c; screenshots to show that regexp.c ran successfully
with KLEE and afl
2. (2 pt) two buggy versions of regexp.c and a readme file that explains where are the bugs and what are the bugs
3. (2 pt) a folder that contains the test inputs generated from AFL and klee
4. (2 pt) a folder that stores the output of running these test inputs on AFL and klee

From Step 5, you'll submit:

1. (3 pt) Source code of your program, modified version for KLEE and AFL (e.g. the harness for testing with these two tools); screenshots to show that the program works with KLEE and afl
2. (1 pt) Explain the known bug(s) contained in the source code that you aim to test
3. (2 pt) a folder that contains the test inputs generated from AFL and KLEE.
4. (2 pt) a folder that stores the output of running these test inputs on AFL and KLEE.

5. (2 pt) Integrate your findings in the summary report

(5 pt) Submit one page report that summarizes your studies. You can use the following questions as a guidance.

(1) How many tests generated by KLEE and AFL respectively?
(2) How many crashes and hangs reported by AFL and KLEE for the 2 programs you
experimented with?
(3) Are these crashes and hangs related to the same bugs or different bugs?
(4) Given a fixed amount of time (e.g., 30 min or 1 hour), which tools find more crashes and
bugs?
(5) Which tools find first crashes and bugs quickly?
(6) What are the advantages and disadvantages of AFL and KLEE? Do you have suggestions to improve the tools?

The homework is due Mar 26, 11:59pm You can continuously submit your homework without a penalty after the deadline. Start early!

References

[1] Docker frequently used commands by Wei Le:
http://weile.blog/2018/11/09/docker-101/