# Project: Compare AFL and KLEE

## 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, we are going to run and further study the testing tools 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.
   1. Download and install AFL
   2. 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.
   1. Easiest way to install KLEE is using the docker image: <https://klee.github.io/docker/>
   2. 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
   1. 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’.
   2. 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/>
   1. 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.
   2. 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.
   3. 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’.
   4. Introduce two buggy versions of regexp.c by implanting two bugs, test regexp.c with both AFL and KLEE.
5. (Extra credit) 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

## Deliverables (20 pt/30 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, modified version for afl;

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?