Assignment 2– Fuzzer Playground (Part 2)

HSS
Fall 2022

In this part, you goal is to enchance AFL++ by adding a feature.

Motivation

One of the common tasks in security research is the ability to modify existing tools for your research. In this part, you will enhance the state-the-art fuzzer, specifically AFL++, to be able to fuzz programs that accept multiple files as their input.

Setup

You will modify the code from the master branch of AFL++ repo.

Repo

`https://github.com/AFLplusplus/AFLplusplus`

Background

AFL++ allows you to fuzz programs that accept a single file as input using its option `-f`. Specifically, consider `strings` program, which takes input as the path to the file.

```
$ strings /usr/bin/cat
```

You need to use the following command to fuzz `strings` using AFL++:

```
$ afl-fuzz -i input -o output -f ./testinput -- strings ./testinput
```

Where, `input` folder contains the initial inputs.
Note: The file name passed to `strings` is same as the one we used for `-f`.

When we specify `-f` option, AFL++ will store its mutated input into that file. Consequently, `strings` runs on the mutated input as the same file is passed as its argument.

Problem

AFL++ supports only a single file; consequently, a program that requires multiple input files cannot be fuzzed effectively.

Goal

In this part, your goal is to modify AFL++, so that it can fuzz programs with multiple input files.

We want to support option `-b` that takes the number of files as the value. AFL++ should generate those many files. The names of these files should have the same prefix as the argument for `-f` and have 1, 2, 3, etc as the suffixes.

For instance, consider that `multiprog` is the program that takes multiple files as input. We can fuzz it using the following command:

```
$ afl-fuzz -i input -o output -f ./testinput -b 3 -- multiprog ./testinput ./testinput1 ./testinput2
```

The above command should generate 3 input files (i.e., `-b 3`) with names starting with `/testinput` (first input file), and `/testinput1` (second input file), `/testinput3` (third input file).

Handling Initial Inputs

You can assume that each file in the initial inputs folder (i.e., `input` folder), is the concatenation of all three files.

Size of the files

All the files should be of the same size. **Hint:** Split the mutated input into `-b` parts and write each part into an input file.

Relevant files

You need to look into the following files to implement this feature.

- Refer `afl-fuzz.c` [1] to add the command line option.
- You can store the number of files info in `afl_state` [2].
- Refer `afl-fuzz-run.c` [3] and `afl-forkserver.c` [4] to split the input into multiple files.

Submission

Use `git diff > afl_modifications.patch` to record the changes made to AFL++ repo and submit the patch file.

[1] https://github.com/AFLplusplus/AFLplusplus/blob/stable/src/afl-fuzz.c#L493
[2] https://github.com/AFLplusplus/AFLplusplus/blob/stable/include/afl-fuzz.h#L415
[3] https://github.com/AFLplusplus/AFLplusplus/blob/stable/src/afl-fuzz-run.c
[4] https://github.com/AFLplusplus/AFLplusplus/blob/stable/src/afl-forkserver.c