AFL Docker Guide

This guide describes how to get started with the afl-docker I provided and experiment with the commands and programs demonstrated in today's discussion. The afl README file can be found here: http://lcamtuf.coredump.cx/afl/README.txt, and the quickstart guide can be found here: http://lcamtuf.coredump.cx/afl/QuickStartGuide.txt. Many of the steps in the quickstart guide have been handled for you by the set up of the docker.

1. Install Docker. I'm not going to get into the details here, but read through the documentation here: https://docs.docker.com/install/ for details on installation and getting started on your system. I believe Prem will go over some of the basics of docker next week and you can come to office hours if you have trouble getting it installed and running.
2. I have stored docker image stored on Dockerhub: https://hub.docker.com/r/caseycas/afl-docker/
   As listed on the page, you can get the docker once docker is installed with "docker pull caseycas/afl-docker". Note that you will need at least a few GBs to download and run the docker.
3. To run the docker, you will need to use "docker run --security-opt seccomp=unconfined -it caseycas/afl-docker". The –security-opt seccomp=unconfined is needed if you want to run gdb inside the docker. This command will open a bash shell in a ubuntu system with afl install and a few examples included.
   a. Note: I have tested the docker on mac and on a windows system, but if you run into issues starting the docker, please let us know.
4. Some important folders in the docker include:
   a. /other-afl-examples/csv/ - This is the root directory for my sample c program with some errors include. I will use this as an example going forward
   b. /guff/ - This an open source project that was used in a tutorial on fuzzing with afl found here: https://spin.atomicobject.com/2015/08/23/fuzz-testing-american-fuzzy-lop/
   c. /afl-2.52b/ - This is the source code for the afl installation.
5. Let's take the example under /other-afl-examples/csv/
   a. We can compile it with:
   "afl-clang -g -o csv_read csv_sample.c"
   afl-clang replaces clang or gcc with a modified compiler that adds instrumentation to the file. I include the -g flag so that the crashes can be investigated with gdb later.
   b. Run the fuzzer with:
   afl-fuzz -i afl/in -o afl/out -- ./csv_read @@
      i. -i is the input directory. -o is the output directory. You need at least one test file in the input directory, but this file cannot be too large or the fuzzer will not work.
      ii. The binary to run is separated with a "—", and the programs work by either taking stdin as input through a pipe, or by taking in the test filenames as command line arguments. To send the test file names as a command line argument (as I do in this example), put an '@@' in the

argument list. This acts as a placeholder for the inputs that the fuzzer will send through.

iii. Once the fuzzer reaches the status screen, let it sit for a few minutes, at least until at least one cycle completes. Detailed information on the status screen can be found here: http://lcamtuf.coredump.cx/afl/status_screen.txt You can exit the status screen with Cntl-c at any time.

c. The structure of the output is explained in the README Section 7) http://lcamtuf.coredump.cx/afl/README.txt

d. If you feed any of the example files from ./afl/out/crashes to the program via (./csv_read <filepath>) you should see a seg fault. If you start the program in gdb and then invoke run <filepath>, gdb should identify the source of the seg fault. Based on my testing with the provided program, you should get at least 8 crashing cases, some failing on the "strlen" call to the program, some falling on the incorrect array access index, and some on the double free.

6. Feel free to try this process on your own C/C++ programs or any open source C/C++ project (as long as they take input from stdin or from files). For c++, you will need to use afl-clang++ or afl-g++ instead of afl-clang to compile. Keep in mind that if you try it on a real application that it can take at the very least hours to complete 1 cycle.

Reference Links:

Docker install and image:
https://docs.docker.com/install/
https://hub.docker.com/r/caseycas/afl-docker/

AFL resources:
http://lcamtuf.coredump.cx/afl/QuickStartGuide.txt
http://lcamtuf.coredump.cx/afl/README.txt
http://lcamtuf.coredump.cx/afl/status_screen.txt

A walkthrough with the "guff" grapher:
https://spin.atomicobject.com/2015/08/23/fuzz-testing-american-fuzzy-lop/
https://github.com/silentbicycle/guff