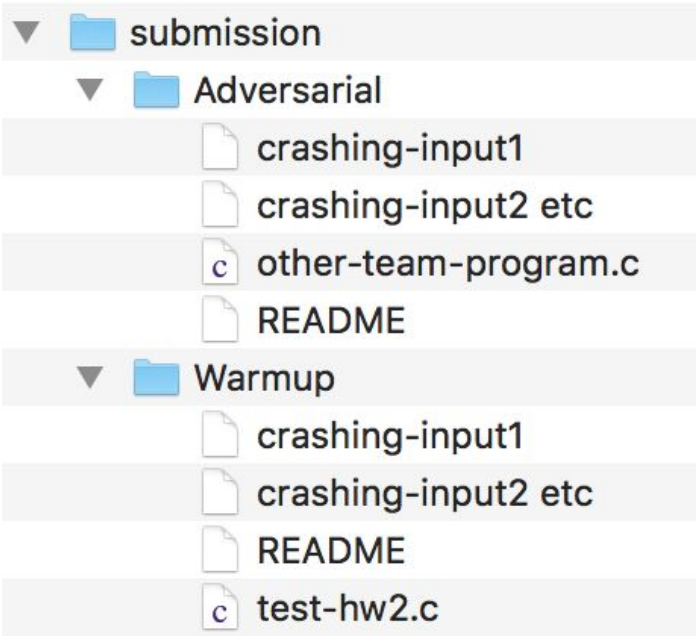
As you will have only 15 minutes for your interactive grading session, we want to streamline the process as much as possible. Following the instructions below will help keep everything moving quickly.

By the Submission Deadline: you should have submitted a set of ***crashing*** ***(not hanging)*** inputs for the example c program the Professor distributed, along with any crashing inputs for the student's submission you are testing. You should submit a ***tar*** file with this structure: (note that other-team-program.c is actually maxTweeter.c that you were assigned to test. The name of the C file doesn't matter, just use the program you were assigned to fuzz test)



The README file should look like this :

Crashing-input1<tab>line-number-where-program-failed
Crashing-input2<tab>line-number-where-program-failed
Etc.

When you come for interactive grading: The grader will pull the version you submitted to canvas to start interactive grading. Also, have a version of the AFL docker running on your terminal before you arrive. Make sure to have this tar file above already set up inside your docker.

Then, the grading session will proceed as follows:

1) First, show a screenshot of the afl running on test-hw2.c. We suggest you save this ahead of time. You don't need to submit it to canvas. You should find several crashes. Your screenshot should show how many distinct crashes you found. Next, show us (within GDB) at least one run, showing where it crashed.
2) Checkout the commit sha of the last commit before the deadline (Sunday) in the repository you are testing from github *inside* the docker. Git is installed on the image already.
    a) git log will let you check the date of the commits.
    b) Use 'git checkout [commit sha] .' to get a specific version of the project.
3) Just show us that the version you checked out (of the adversary program) is the same as the one in the submission above
4) Show the screenshot of the AFL status screen from when you ran it (you will not be rerunning the AFL in the session).
5) If you've found at least 5 distinct bugs in the program you will get 10 points from the other team, so you do not need to show us more than that. To demonstrate each bug, do the following steps:
    a) Compile the submitted program.
    b) Start the program in gdb.
    c) Run the program with the crashing input and demonstrate a crash.
    d) If gdb reports an error on a distinct line, then you get 2 points for the crash.
    e) If gdb reports two inputs as error causing that crash on the same line, *and you can conclusively show that these are distinct errors*, then you will get credit for both. Credit here is at the discretion of the grader. Prepare, short, concise arguments as to why they are distinct. For each distinct bug, you'll have a maximum of 2 minutes to make your argument.