Summarize the work completed so far

We educated ourselves on the basic techniques of fuzzing, demonstrated by cloning the github repository, installing the toolchain, and building the program to test the programs given in the paper. Additionally, we read multiple papers regarding different parallel implementations of AFL fuzzing, understanding what kind of goals they were trying to achieve and the performance of each. Now we are attempting to tweak the AFL repository to make a parallel version ourselves, and see if that performs in a comparable fashion to other parallel implementations of AFL.

Goals and Deliverables

We believe that it will be difficult to implement the program by the deadline. The difficulty that we are facing is that none of the papers that we are referencing provide the code sample that they claimed to have built. We do not have a reference code base to start from other than the AFL fuzzing code base, but that codebase is massive and difficult to understand. However, we do believe that once we implement a working version, testing the performance of such will still be viable.

Poster Session

At the poster session, we plan on presenting graphs on performances that we gathered.

Preliminary Results

We do not have any at the moment.

Issues

We sort of hit a dead end - there is a huge gap between the AFL fuzzing code base and the final parallel implementation of AFL fuzzing since modifying the code base requires extensive knowledge of how the AFL fuzzing platform works alongside the OS and also a lot of time writing correct code and debugging it. We hoped in the beginning to find some reference parallel version so that the only tweak we would need to do is the sync algorithm and the frequency of synchronization. However, we were devastated to find that none of the authors of the papers that claimed to have achieved higher performance than naive parallel fuzzing have their source code up online. With our current knowledge of parallel systems and AFL, analyzing the code base of AFL wasn’t too helpful either.

New Schedule

Week 1 (November 1 - November 7)

- ~~Work on shaping out the overall project, and write the proposal~~
- ~~Read and understand the paper we will be using, as well as other papers on fuzzing~~
- ~~Read through the documentation and code of AFL++ to understand their implementation of program states~~

Week 2 (November 8 - November 14)

- ~~Read more through the documentation and code of AFL++~~
- ~~Try compiling and running AFL in vanilla parallel mode~~

Week 3 (November 15 - November 21)

- ~~Start working on implementing PAFL~~
- ~~Write Milestone Report; due on November 22, 9 am~~

Week 4, 1/2 (November 22 - November 25) <= we are here

- Continue working on PAFL

Week 5, 1/2 (November 29 - December 2)

- Finish up implementation
- Start testing for performance

Week 6 (December 6 - December 9)

- Finish up testing for performance
- Write Final Project Report; due on December 9, 11:59 pm
- Prepare for Project Poster Session

Project Poster Session (December 10)

- Look at other people's projects
- Have fun