For this assignment, I found a repository called Star Ruler 2, and in this repository is a str_util.cpp file. This file contains methods to manipulate strings, e.g., replace, split, and join. I decided to test the methods in this file because they are pretty straight-forward and easy to fuzz. To test these methods, I created one big GTest called “StrManipulation”. I tested each individual method first to make sure the output was the same as std::string output. I then created a OneOf, so that I could test a random number of methods calls on a string. I also created helper methods that compare the toUpper, toLower, and replace methods with equivalent c methods. I used the DeepState method DeepState_CStr_C to create a string from these characters:
“abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_()/?*&%$#@![]{}”. I wanted to see if any special characters would cause any unwanted behavior.

This package contains split/join methods to split a string into a vector, and then join that vector into a string. I would assume that you would be able to do a roundtrip and get the original string, but this does not seem the case. The test fails every time on any input. My guess is that I may be splitting on a char that is not in the string, but this should not cause any problems with the join. The method may not handle splitting on a character not in the string. I then commented the code out so that I could test the other methods.

I first used the deepstate angr fuzzer without a seed. It ran 24757 tests with 2 failing tests. My first run was with –fuzz, but this caused my output directory to be empty. The only way I could get failure to save was to save all output. The second time I ran it, I got 20480 passed tests and 1 failure. The end of the failure test’s output is:

```
TRACE: Tests.cpp(96): Number of Iterations: 9
TRACE: Tests.cpp(115): Started Trim: gtcg73eRReg]erkejsqf)ztG
TRACE: Tests.cpp(118): Trimmed whitespace
TRACE: Tests.cpp(96): Number of Iterations: 10
TRACE: Tests.cpp(128): Started Replace: gtcg73eRReg]erkejsqf)ztG
TRACE: Tests.cpp(143): Replaced substring: ]erkejsqf) with 4]2e@8TMYw string:gtcg73eRReg4]2e@8TMYwztG
TRACE: Tests.cpp(96): Number of Iterations: 11
TRACE: Tests.cpp(128): Started Replace: gtcg73eRReg4]2e@8TMYwztG
TRACE: Tests.cpp(143): Replaced substring: ]2e@8TMYwzt with r4x?XYMJT[K string:gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(96): Number of Iterations: 12
TRACE: Tests.cpp(121): Started Char Trim: gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(125): Trimmed char:  string: gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(96): Number of Iterations: 13
TRACE: Tests.cpp(121): Started Char Trim: gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(125): Trimmed char:  string: gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(96): Number of Iterations: 14
TRACE: Tests.cpp(99): Started Lowercase: gtcg73eRReg4r4x?XYMJT[KG
TRACE: Tests.cpp(104): Our Lowercase: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(96): Number of Iterations: 15
TRACE: Tests.cpp(115): Started Trim: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(118): Trimmed whitespace
TRACE: Tests.cpp(96): Number of Iterations: 16
TRACE: Tests.cpp(99): Started Lowercase: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(104): Our Lowercase: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(96): Number of Iterations: 17
TRACE: Tests.cpp(115): Started Trim: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(118): Trimmed whitespace
TRACE: Tests.cpp(96): Number of Iterations: 18
TRACE: Tests.cpp(115): Started Trim: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(118): Trimmed whitespace
TRACE: Tests.cpp(96): Number of Iterations: 19
TRACE: Tests.cpp(121): Started Char Trim: gtcg73erreg4r4x?xymjt[kg
TRACE: Tests.cpp(125): Trimmed char: g string: tcg73erreg4r4x?xymjt[k
TRACE: Tests.cpp(96): Number of Iterations: 20
TRACE: Tests.cpp(128): Started Replace: tcg73erreg4r4x?xymjt[k
terminate called after throwing an instance of 'std::out_of_range'
  what():  basic_string::substr: __pos (which is 23) > this->size() (which is 22)
ERROR: Failed: StrUtil_StrManipulation
ERROR: Test case angr_out/1e0bf949343f624237a2309124f034052101a4ae.fail failed
```

The output shows that the test failed is in the oneOf when tying to get a substring of a string. To get a substring, I am using std::string::substr. This is not a problem with my methods, but something that happened when picking indices for the substring to replace. Not 100% sure why this error occurred and why it occurs a very small amount of the time.

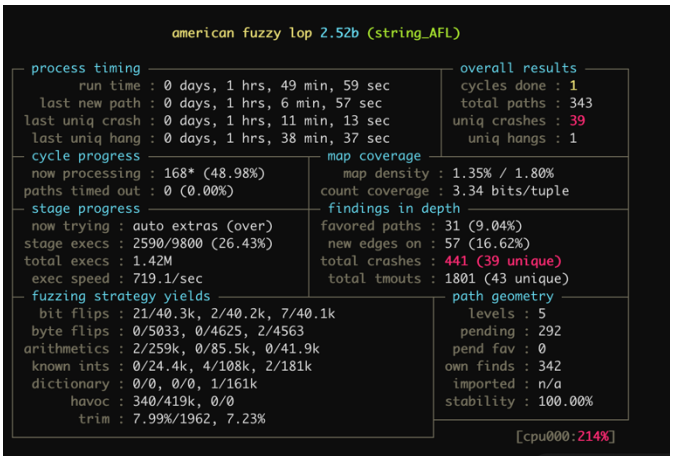
I decided to rerun the tests to see if something different happens, this time with –exit_on_fail. A crash occured and the end of the output is:

```
TRACE: Tests.cpp(96): Number of Iterations: 12
TRACE: Tests.cpp(146): Started Replace Char: }Dm~-T4l*45 1mpU0j]E$N
TRACE: Tests.cpp(151): Replaced char: 42 with: 92 string: }Dm~-T4l\45 1mpU0j]E$N
TRACE: Tests.cpp(96): Number of Iterations: 13
TRACE: Tests.cpp(128): Started Replace: }Dm~-T4l\45 1mpU0j]E$N
TRACE: Tests.cpp(143): Replaced substring: ~-T with 7Tc string:}Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(96): Number of Iterations: 14
TRACE: Tests.cpp(146): Started Replace Char: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(151): Replaced char: 47 with: 61 string: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(96): Number of Iterations: 15
TRACE: Tests.cpp(146): Started Replace Char: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(151): Replaced char: -69 with: -16 string: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(96): Number of Iterations: 16
TRACE: Tests.cpp(115): Started Trim: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(118): Trimmed whitespace
TRACE: Tests.cpp(96): Number of Iterations: 17
TRACE: Tests.cpp(107): Started Uppercase: }Dm7Tc4l\45 1mpU0j]E$N
TRACE: Tests.cpp(112): Our Uppercase: }DM7TC4L\45 1MPU0J]E$N
TRACE: Tests.cpp(96): Number of Iterations: 18
TRACE: Tests.cpp(128): Started Replace: }DM7TC4L\45 1MPU0J]E$N
```

This test crashed in the same spot, but this time a timeout occurred, not a failure. The timeout occurred in the replace() function we are testing. Again, I am not sure why this test timed out.

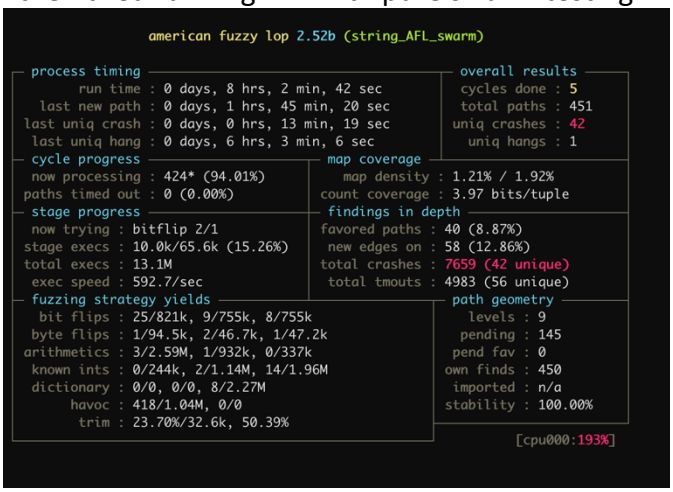
To end my angr testing, I decided to try pure swarm testing. This resulted in 5851 passing tests and 1 failure. The failure occurred for the same reason as the failure above. The output would not save though.

I then ran DeepState with AFL and used the same string as above as the input. The output is:



The AFL test ran for almost 2 hours (without -d), completing one cycle and finding 343 paths. I was not surprised by the path count since there is a lot of randomization in the test, i.e., replacing a random substring each time. However, I was surprised by the number of failures and timeouts. There was a total of 441 crashes, 39 unique, and 1801 timeouts, 43 unique. I reran some of the tests, and they are either timeouts or failures because of the std::out_of_range exception.

I then tried running AFL with pure swarm testing. This resulted in the following:

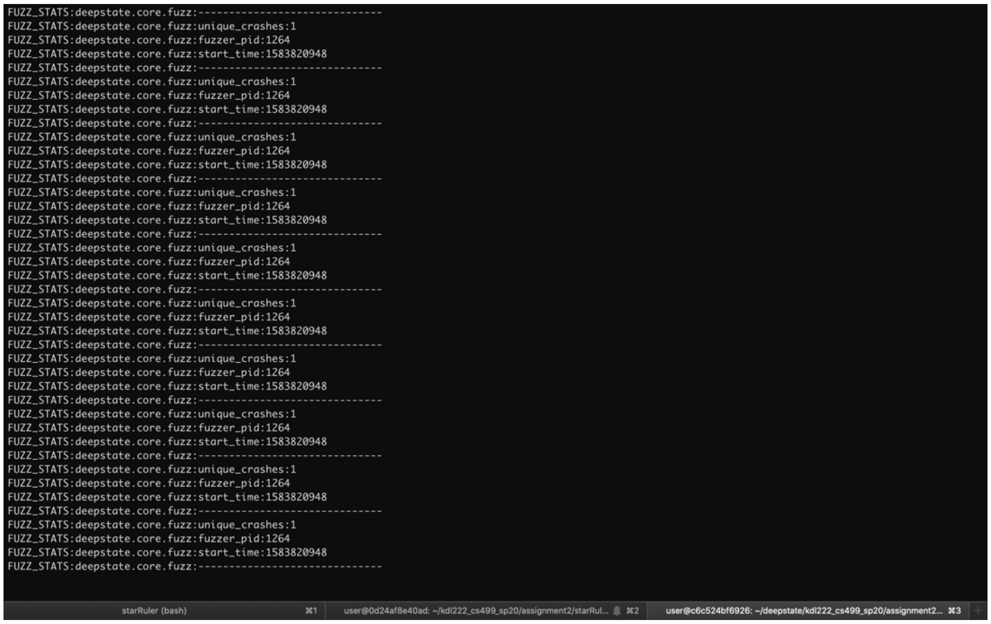


I let AFL run for 8 hours without the -d flag. It went through 5 cycles and found 451 paths. To my surprise, 42 unique crashes and 56 unique timeouts occurred, but the total number of crashes and timeouts is very large. As I watched it at the beginning, I noticed that it also took a little longer for AFL to find the crashes. This could have been because the replace method was being called less (not being used a lot at the beginning).

I tried using LibFuzzer but could not get it to work. If I used deepstate-libfuzzer, I got a python arg parse error. I tested this with the RunLen DeepState example. I also tried running the executable directly, but it also failed to run with “Killing the process and childs.”

I alsocould not get Eclipser to run either. When I tried to run my executable with deepstate-eclisper, I get an error about my output directory not being there, when it clearly is. I tried to give a full path, but this did not fix the problem.

In order to use HonggFuzz, I had to use the new DeepState Docker container. I let the fuzzer run for almost 9 hours. The end of the output is below:



HonggFuzz found 1 unique crash. After replaying the crash, I found that the crash was caused by the same std::string std::out_of_range error.

Overall, I am not sure if the methods I tested caused any of the failures, but if how I wrote the tests caused them. Most of the time, they pass, but sometimes a std::out_of_range exception occurs or a timeout occurs in the replace() method. However, the timeout may actually be a real problem.