In a new build folder, do the below for 5 rounds

CC=/usr/local/bin/afl-gcc CXX=/usr/local/bin/afl-g++ cmake ..

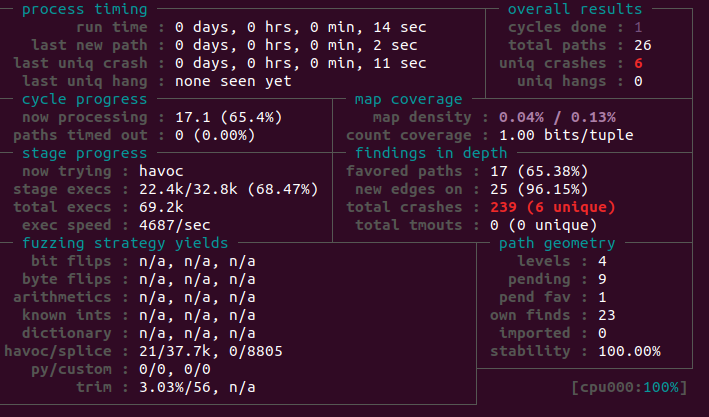
make

afl-fuzz -i ../testcases -o ../findings ./afldemo

rm -rf ./

**Round 1:**

For part 1:**Fuzz testing on a vulnerable program**

****

*(1) how long was the fuzzing process*

14 sec

*(2) how many inputs AFL found that could make the program crash or hang.*

6

wenhui@wenhui:~/Downloads/afl-demo/findings/default/crashes$ ls

id:000000,sig:06,src:000002,time:80,op:havoc,rep:8

id:000001,sig:11,src:000002,time:147,op:havoc,rep:2

id:000002,sig:06,src:000002,time:1302,op:havoc,rep:4

id:000003,sig:11,src:000002,time:1317,op:havoc,rep:16

id:000004,sig:06,src:000002,time:1585,op:havoc,rep:16

id:000005,sig:06,src:000002,time:3449,op:havoc,rep:4

For part 2: **Isolating and fixing the bugs**

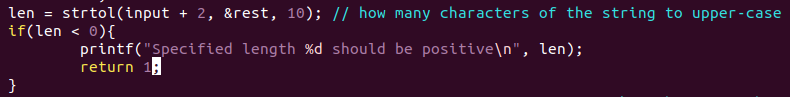
*(1) describe your process of identifying bugs from AFL-generated inputs and what bugs you found in the program;*

The AFL inputs are in folder /findings/default/crashes, execute the program with this input , and use printf to identify the issue,

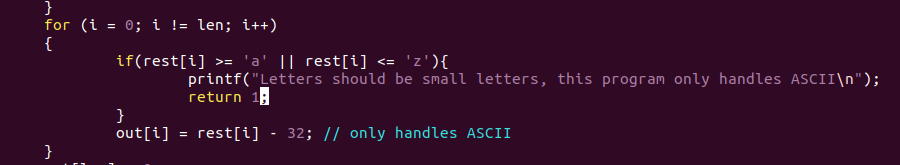
1. u -1 strin, It seems like N should not be a negative number, it causes malloc errors
2. u -18 aBCd 18 aBC11, it seems like negative N causes malloc issues
3. u 11 {BCdddd@ddddMddddddd0aB9zde, it seems like characters, such as {, other than small letters causes transiting issues
4. u 10 aBCded 10 aBCde0 aBC aBCd 10 aBCde0 aBCd, it seems like spaces and capital letters causes translating issues
5. u 0aBCu 8 4 aBCde, it seems like spaces and capital letters causes translating issues
6. u 0aBCu 8 4 aBCde aBCdeed����, it seems like spaces and capital letters causes translating issues

*(2) describe how you fixed the bugs and why your fixes worked.*

For INPUT in case a and b: Bound the input N to positive integers, bound the N so that later malloc won’t have issues

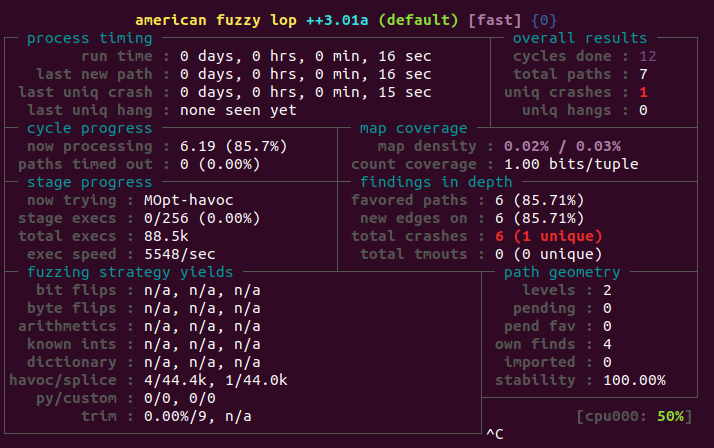


For INPUT in case c, d, e, f, Bound the input string to only small letters, so that ASCII translation works



**Round 2:**

For part 1:**Fuzz testing on a vulnerable program**

****

*(1) how long was the fuzzing process;*

16 sec

*(2) how many inputs AFL found that could make the program crash or hang.*

**

*1 input for crash:*

*u u �� �� 3 � >B*

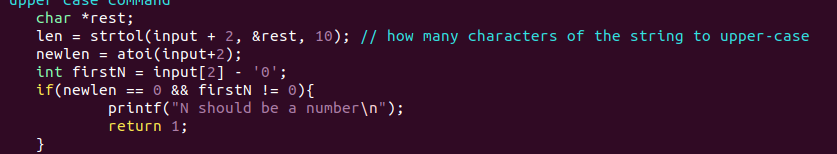
For part 2: **Isolating and fixing the bugs**

*(1) describe your process of identifying bugs from AFL-generated inputs and what bugs you found in the program;*

Input “u u �� �� 3 � >B” , execute the program with this input , and use printf to identify the issue, N should be an integer instead of a character

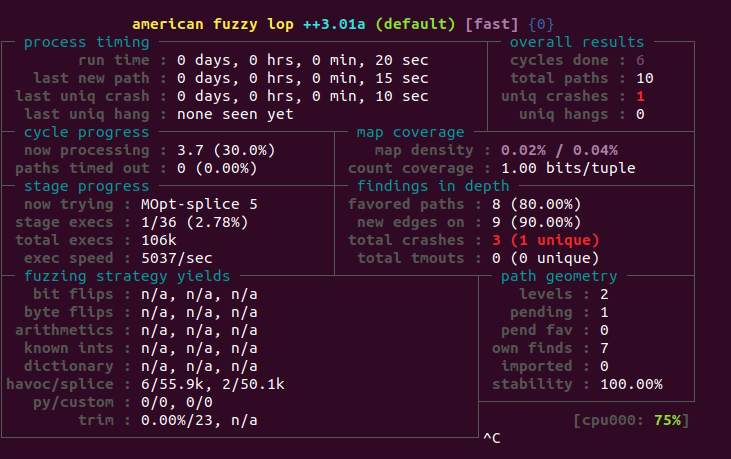
*(2) describe how you fixed the bugs and why your fixes worked.*

*Bound N to a number, atoi returns 0 if it is a number 0 or not a number. Get the first char to see if the N is 0, if N is not 0 and atoi returns 0, then the input is not valid.*

**

**Round 3:**

For part 1:**Fuzz testing on a vulnerable program**

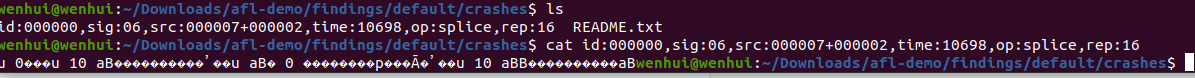
****

*(1) how long was the fuzzing process;*

20 sec

*(2) how many inputs AFL found that could make the program crash or hang.*

1 input



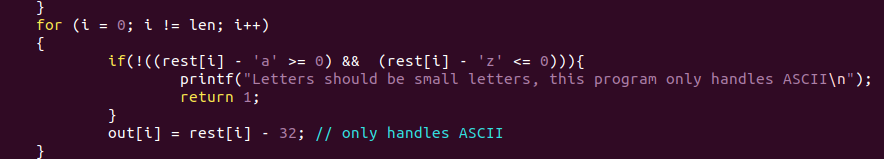
For part 2: **Isolating and fixing the bugs**

*(1) describe your process of identifying bugs from AFL-generated inputs and what bugs you found in the program;*

Input “u 0���u 10 aB����������ߴ��u aB� 0 ��������p���Ā�ߴ��u 10 aBB����������aB” , execute the program with this input , and use printf to identify the issue, input string should be small letters

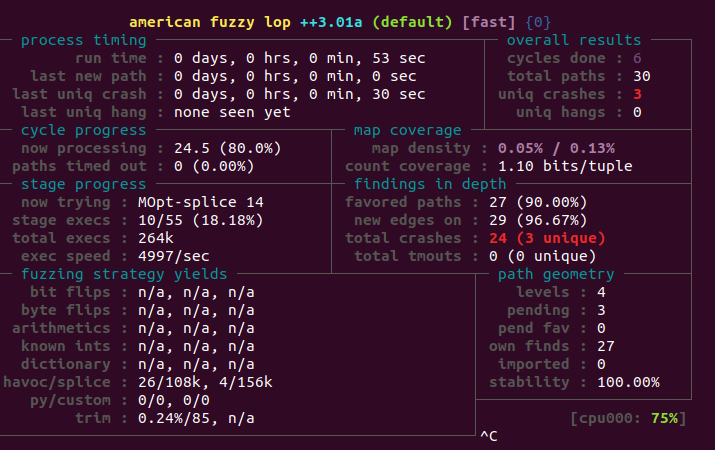
*(2) describe how you fixed the bugs and why your fixes worked.*

*All characters in string must be between ‘a’ and ‘z’.*

**

**Round 4:**

For part 1:**Fuzz testing on a vulnerable program**

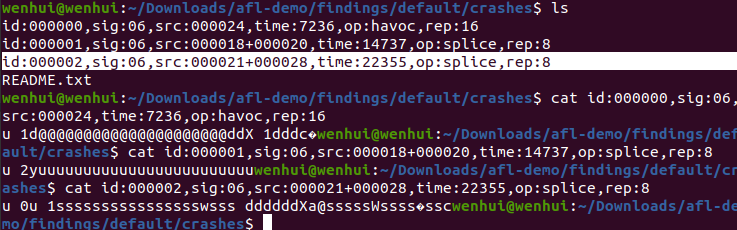
****

*(1) how long was the fuzzing process;*

53 sec

*(2) how many inputs AFL found that could make the program crash or hang.*

*3*



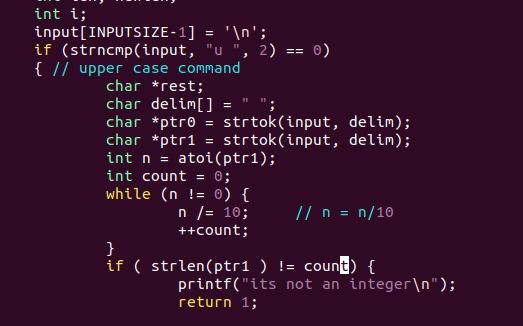
For part 2: **Isolating and fixing the bugs**

*(1) describe your process of identifying bugs from AFL-generated inputs and what bugs you found in the program;*

execute the program with this input , and use printf to identify the issue, there should be a space after the N, also the input should terminated with \n, there should be a space after N

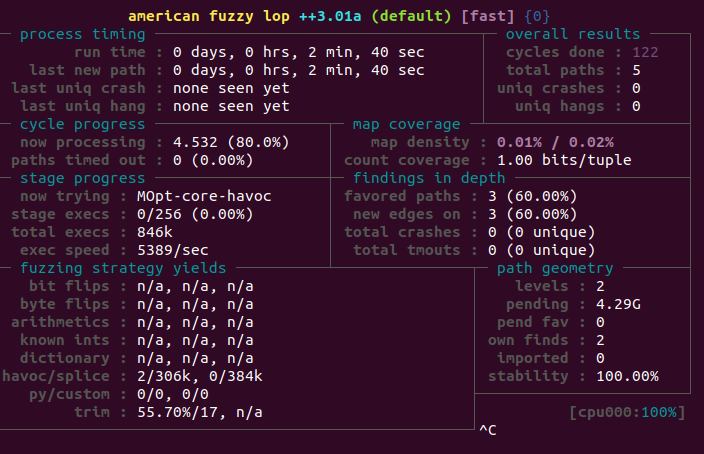
*(2) describe how you fixed the bugs and why your fixes worked.*

*Compare the length of string N with digit N, if they are the same length, then it is an integer, otherwise it is not an integer*

**

**Round 5:**

For part 1:**Fuzz testing on a vulnerable program**

****

*(1) how long was the fuzzing process;*

*2 mins 40 sec*

*(2) how many inputs AFL found that could make the program crash or hang.*

none

For part 2: **Isolating and fixing the bugs**

*(1) describe your process of identifying bugs from AFL-generated inputs and what bugs you found in the program;*

*N/A*

*(2) describe how you fixed the bugs and why your fixes worked.*

*N/A*