

Assignment 4 report

Yan Duan & Qinhuai Xu

2. PasswordCheck.c for KLEE

B) Run KLEE on the modified program, with default options

i) How many bugs are detected by KLEE? Explain the nature of all detected bug(s). If not, explain why no bugs were found.

- There is 1 bug in detected by KLEE, which is invalid klee_assume call. The bug is triggered because the assertion (`pwdChar >= 'a' && pwdChar <= 'z'`) is evaluated as false in running.

```
klee@bbe8d588b824:~/work$ vim KLEE1.c
klee@bbe8d588b824:~/work$ clang -emit-llvm -S -c KLEE1.c -o KLEE1.ll
klee@bbe8d588b824:~/work$ klee KLEE1.ll
KLEE: output directory is "/home/klee/work/klee-out-1"
KLEE: Using STP solver backend
KLEE: SAT solver: MiniSat
KLEE: WARNING: undefined reference to function: printf
KLEE: ERROR: (location information missing) invalid klee_assume call (provably false)
KLEE: NOTE: now ignoring this error at this location
KLEE: WARNING ONCE: calling external: printf(93900321227264) at [no debug info]
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
```

ii) How many total paths were explored by KLEE?

- There are totally 4096 completed paths and 4095 partially completed paths were explored by KLEE.

```
KLEE: done: total instructions = 262541
KLEE: done: completed paths = 4096
KLEE: done: partially completed paths = 4095
KLEE: done: generated tests = 4097
klee@bbe8d588b824:~/work$
```

iii) Approximately, how long did KLEE take to run?

- 2.08 seconds.

```
klee@bbe8d588b824:~/work$ klee-stats klee-out-1
-----
| Path | Instrs | Time(s) | ICov(%) | BCov(%) | ICount | TSolver(%) |
-----
|klee-out-1| 262541| 2.08 | 100.00 | 100.00 | 139 | 41.18 |
-----
```

iv) Why is KLEE considered a concolic execution tool rather than a pure symbolic execution tool? What are the differences?

- Concolic execution is a combination of both concrete (real-world values) and symbolic (variables represented as symbols) execution, whereas pure symbolic execution relies solely on symbolic values.
- KLEE is considered as a concolic execution tool because it combines concrete execution and

symbolic execution in its analysis. KLEE starts its analysis by executing the program using concrete input values. As the program executes, KLEE collects symbolic constraints on the input variables by symbolic execution.

C) Now apply a sanitizer, by passing in *fsanitize=signed-integer-overflow* when building KLEE1.c
 i) What is the effect on KLEE by applying this sanitizer, compared to your previous observations without sanitizers? Is KLEE able to find any new bugs? Does KLEE explore additional paths? Does KLEE take longer to run?

- Compared to previous observations without sanitizers, the number of total instructions is larger, with 451043 than 262541.
- KLEE does not find any new bugs.
- KLEE does not explore additional paths.
- KLEE takes longer to run than previous observation, with 3.30 seconds.

```
klee@bbe8d588b824:~/work$ clang -fsanitize=signed-integer-overflow -emit-llvm -S
-c KLEE1.c -o KLEE1.ll
klee@bbe8d588b824:~/work$ klee KLEE1.ll
KLEE: output directory is "/home/klee/work/klee-out-2"
KLEE: Using STP solver backend
KLEE: SAT solver: MiniSat
KLEE: WARNING: undefined reference to function: __ubsan_handle_add_overflow
KLEE: WARNING: undefined reference to function: __ubsan_handle_sub_overflow
KLEE: WARNING: undefined reference to function: printf
KLEE: ERROR: (location information missing) invalid klee_assume call (provably false)
KLEE: NOTE: now ignoring this error at this location
KLEE: WARNING ONCE: calling external: printf(94745183360672) at [no debug info]
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
Password did not match
```

```
Password did not match
Password did not match
Password did not match
Password did not match

KLEE: done: total instructions = 451043
KLEE: done: completed paths = 4096
KLEE: done: partially completed paths = 4095
KLEE: done: generated tests = 4097
```

```
klee@bbe8d588b824:~/work$ klee-stats klee-out-2
```

Path	Instrs	Time(s)	ICov(%)	BCov(%)	ICount	TSolver(%)
klee-out-2	451043	3.30	92.96	80.00	199	51.61

ii) How does a sanitizer work?

The sanitizer works by adding checks to detect specific problematic condition. For example, when you enable *fsanitize=signed-integer-overflow*, the compiler adds checks to the program's signed integer operations. During program execution, the instrumented code will perform runtime checks to detect signed integer overflows. If an overflow occurs, the sanitizer will intercept it and generate an alert and report relevant information.

B) Run AFL on the modified program, with default options. You must supply an input folder with seed files. Start with minimal number of seed files.

i) How many crashes and hangs were encountered by AFL?

- There are 4 crashes, and 0 hangs were encountered by AFL.

```
american fuzzy lop ++3.15a [default] (./AFL1) [fast]
process timing
  run time      : 0 days, 0 hrs, 9 min, 2 sec
  last new find : none seen yet
last saved crash : 0 days, 0 hrs, 9 min, 0 sec
last saved hang  : none seen yet
cycle progress
  now processing : 1.6 (20.0%)
  runs timed out : 0 (0.00%)
stage progress
  now trying      : splice 14
  stage execs     : 33/220 (15.00%)
  total execs     : 21.8k
  exec speed      : 20.54/sec (slow!)
fuzzing strategy yields
  bit flips       : disabled (default, enable with -D)
  byte flips      : disabled (default, enable with -D)
  arithmetics     : disabled (default, enable with -D)
  known ints      : disabled (default, enable with -D)
  dictionary       : n/a
havoc/splice      : 4/9214, 0/12.6k
py/custom/rq      : unused, unused, unused, unused
  trim/eff         : 7.69%/3, disabled
map coverage
  map density      : 0.00% / 0.00%
  count coverage   : 1.00 bits/tuple
findings in depth
  favored items    : 2 (40.00%)
  new edges on    : 2 (40.00%)
  total crashes    : 4 (4 saved)
  total tmoouts    : 9516 (5 saved)
item geometry
  levels          : 1
  pending         : 1
  pend fav        : 1
  own finds       : 0
  imported        : 0
  stability        : 100.00%
[cpu000:116%]
```

ii) How many bugs are detected by AFL? Explain the nature of all detected bug(s). If not, explain why no bugs were found.

- To find bugs, we try to check crashes files and find the crash is triggered by “asdfgzc” input. Because the length of “asdfgzc” is 8, which is less than the defined SIZE 12 in line 5 and cause password array index out of bound bug in line 14.

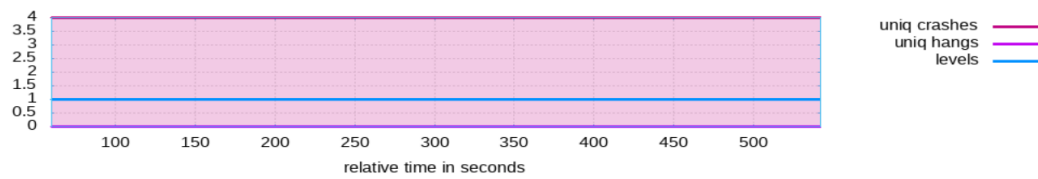


```
5      #define SIZE 12

13     for (int i=0; i < SIZE; i++) {
14         passwordBuffer[i] = password[i];
15     }
```

iii) How long did AFL take to encounter its first crash or hang, if one was ever found?

- AFL encountered its first crash very soon, in a few seconds.



C) Now apply sanitizers to AFL, by passing in `fsanitize=signed-integer-overflow`, address, undefined

i) How many crashes and hangs were detected by AFL, with sanitizers, compared to the program without sanitizers?

- 1 crash and 1 hang were detected by AFL with sanitizer `signed-integer-overflow`

```

File Edit View Search Terminal Help
cpen@cpenUBC-VirtualBox: ~/AFLplusplus/integer

american fuzzy lop ++3.15a {default} (./AFL1) [fast]
- process timing
  run time : 0 days, 0 hrs, 18 min, 42 sec
  last new find : none yet (odd, check syntax!)
  last saved crash : 0 days, 0 hrs, 18 min, 42 sec
  last saved hang : 0 days, 0 hrs, 16 min, 2 sec
- cycle progress
  now processing : 1.9 (20.0%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : splice 8
  stage execs : 78/110 (70.91%)
  total execs : 19.8k
  exec speed : 20.19/sec (slow!)
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 4/8496, 0/11.2k
  py/custom/rq : unused, unused, unused, unused
  trin/eff : 11.11%/4, disabled

overall results
cycles done : 1
corpus count : 5
saved crashes : 1
saved hangs : 1

map coverage
map density : 0.00% / 0.00%
count coverage : 1.00 bits/tuple

findings in depth
favored items : 2 (40.00%)
new edges on : 2 (40.00%)
total crashes : 1 (1 saved)
total tmouts : 8869 (5 saved)

item geometry
levels : 1
pending : 0
pend fav : 0
own finds : 0
imported : 0
stability : 100.00%

[cpu002: 66%]

```

- 4 crashes and 1 hang were detected by AFL with sanitizer `address`

```

File Edit View Search Terminal Help
cpen@cpenUBC-VirtualBox: ~/AFLplusplus/address

american fuzzy lop ++3.15a {default} (./AFL1) [fast]
- process timing
  run time : 0 days, 0 hrs, 16 min, 50 sec
  last new find : none seen yet
  last saved crash : 0 days, 0 hrs, 16 min, 47 sec
  last saved hang : 0 days, 0 hrs, 10 min, 34 sec
- cycle progress
  now processing : 1.5 (20.0%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : havoc
  stage execs : 909/1766 (51.47%)
  total execs : 15.5k
  exec speed : 32.34/sec (slow!)
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 4/8396, 0/8132
  py/custom/rq : unused, unused, unused, unused
  trin/eff : 11.11%/4, disabled

overall results
cycles done : 0
corpus count : 5
saved crashes : 4
saved hangs : 1

map coverage
map density : 0.00% / 0.00%
count coverage : 1.00 bits/tuple

findings in depth
favored items : 2 (40.00%)
new edges on : 2 (40.00%)
total crashes : 4 (4 saved)
total tmouts : 5985 (6 saved)

item geometry
levels : 1
pending : 0
pend fav : 0
own finds : 0
imported : 0
stability : 100.00%

[cpu001:150%]
^C

```

- 3 crashes and 0 hang were detected by AFL with sanitizer `undefined`

```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/undefined
File Edit View Search Terminal Help

american fuzzy lop ++3.15a [default] (./AFL1) [fast]
- process timing
  run time : 0 days, 0 hrs, 21 min, 20 sec
  last new find : 0 days, 0 hrs, 20 min, 31 sec
  last saved crash : 0 days, 0 hrs, 21 min, 19 sec
  last saved hang : none seen yet
- cycle progress
  now processing : 7.37 (63.6%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : splice 14
  stage execs : 9/36 (25.00%)
  total execs : 36.1k
  exec speed : 12.75/sec (zzzz...)
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 9/14.1k, 0/21.9k
  py/custom/rq : unused, unused, unused, unused
  trln/eff : 47.11%/27, disabled
- overall results
  cycles done : 3
  corpus count : 11
  saved crashes : 3
  saved hangs : 0
- map coverage
  map density : 0.00% / 0.00%
  count coverage : 1.77 bits/tuple
- findings in depth
  favored items : 2 (18.18%)
  new edges on : 3 (27.27%)
  total crashes : 4 (3 saved)
  total tmouts : 15.9k (4 saved)
- item geometry
  levels : 3
  pending : 3
  pend fav : 0
  own finds : 6
  imported : 0
  stability : 100.00%
[cpu000:166%]
^C

```

ii) How many bugs are detected by AFL? Explain the nature of all detected bug(s). If not, explain why no bugs were found.

- We try to check the crashes files and they are all same issues as previous. Therefore, no new bugs are detected by AFL.

iii) Try modifying the seed input files, and/or adding more input seed files. Does this change the AFL results in any way?

- We try to add 3 more input seed files, which is seed_file2.txt, seed_file3.txt, and seed_file4.txt.
 - seed_file2.txt contains characters.
 - seed_file3.txt contains characters and digits.
 - seed_file4.txt contains more special characters.

```

Open [ ] seed_file2.txt ~/AFLplusplus/modifiedSeed/input Save [ ] [ ] [ ]
abcdefghijklmnopqrstuvwxyz

```

```

Open [ ] seed_file3.txt ~/AFLplusplus/modifiedSeed/input Save [ ] [ ] [ ]
cas12ew3

```

```

Open [ ] seed_file4.txt ~/AFLplusplus/modifiedSeed/input Save [ ] [ ] [ ]
fweq3*^saf%3ca2@

```

- After adding more seeds, 5 crashes and 0 hang were detected by AFL with sanitizer signed-integer-overflow.

detected by AFL.

iv) Having used both KLEE and AFL on the same program, which tool do you find more effective in finding bugs in this scenario? What are the advantages and disadvantages of using each tool respectively?

- We find that KLEE is more effective in finding bugs on *PasswordCheck.c* because KLEE takes less running time than AFL.
- KLEE:
 - Pros: Because KLEE is a concolic execution tool, it can trigger the bug with precise input values, which help developers quickly identifying and fixing issues.
 - Cons: KLEE's focus on path exploration but it may not be able to handle system calls, and external dependencies effectively like AFL.
- AFL:
 - Pros: AFL can handle system calls, and external dependencies more effectively than KLEE.
 - Cons: AFL heavily relies on random mutations of inputs, which may lead to limited path coverage. Also, the large number of test inputs will lead to longer running time.

4. Vulnerable.c for KLEE

B) Run KLEE on the modified program, with the (uClibc) C standard library enabled

i) Explain your observations from using KLEE on this program.

- There are 4 bugs found, including invalid pointer type bugs and out of bound pointer type bugs.
- There are 101 partially completed paths and 0 completed path.
- 1.56 seconds were taken.

```

Input:
size : 0
Input:
size : 2
Input:
Input:
KLEE: ERROR: KLEE2.c:34: memory error: invalid pointer: free
KLEE: NOTE: now ignoring this error at this location
size : 6
size : 4
Input:
Input:
Input:
size : 8
KLEE: ERROR: KLEE2.c:37: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
size : 10
size : 12
Input:
Input:
size : 14
size : 16
Input:
Input:
size : 18
Input:
size : 20
Input:
size : 22
size : 24
KLEE: ERROR: libc/string/memcpy.c:29: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
Input:
Input:
size : 26
Input:
size : 28
size : 30
KLEE: ERROR: libc/string/strlen.c:22: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
Input:
Input:
size : 196
size : 198

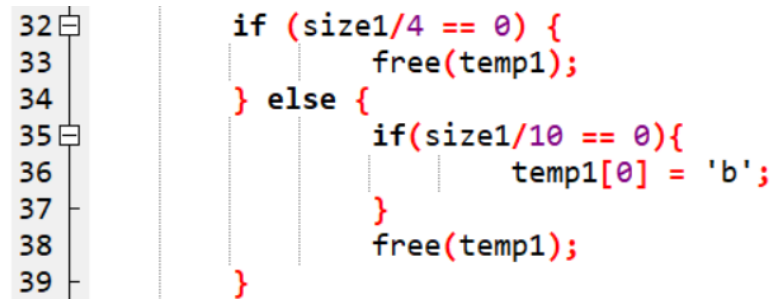
KLEE: done: total instructions = 180447
KLEE: done: completed paths = 0
KLEE: done: partially completed paths = 101
KLEE: done: generated tests = 4
klee@bbe8d588b824:~/work$

klee@bbe8d588b824:~/work$ klee-stats klee-out-6
-----
| Path | Instrs| Time(s)| ICov(%)| BCov(%)| ICount| TSolver(%)|
-----
|klee-out-6| 182709| 1.56| 40.35| 26.45| 2823| 58.57|
-----

```

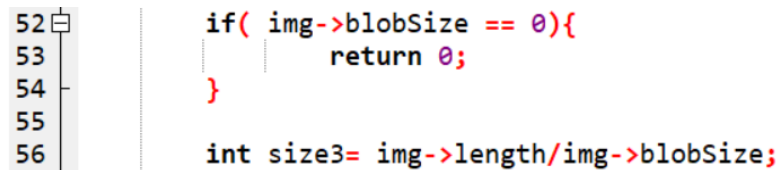
ii) Try fixing a few bug(s) in the program, and recompiling the program again. Then, run KLEE on it again. Do you discover any new bugs or paths?

- Bug 1: free(temp1)
Move “free(temp1)” on line 31 into the if sentence (line 38) to avoid being free twice.



- Bug 2: 0 as divisor

Check blobSize value before division to avoid 0 as divisor.



- After fixing the above 2 bugs and recompiling the program, we run KLEE on it again. We find 1 new bug and 1 remaining bug. There are 1 completed path and 63 partially completed paths.

```
Input:
size : 0
Input:
Input:
size : 2
size : 4
Input:
Input:
size : 6
Input:
size : 10
Input:
size : 8
size : 12
Input:
size : 14
Input:
KLEE: ERROR: KLEE2.c:66: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
Input:
size : 16
Input:
size : 18
KLEE: ERROR: libc/string/memcpy.c:29: memory error: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
size : 20
Input:
size : 22
Input:
size : 24
Input:
Input:
size : 26
Input:
size : 30
size : 28
Input:
Input:
Input:
```

```

KLEE: done: total instructions = 86668
KLEE: done: completed paths = 1
KLEE: done: partially completed paths = 63
KLEE: done: generated tests = 3

```

5. Vulnerable.c for AFL

B) Run AFL on the modified program

i) Explain your observations from using AFL on this program and try different sanitizers

- 0 crash and 0 hang were detected by AFL without sanitizer

```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/vulnerable
File Edit View Search Terminal Help

american fuzzy lop ++3.15a {default} (./AFL2) [fast]
┌──────────┴──────────┐
┌ process timing ────┐ ┌ overall results ────┐
│ run time : 0 days, 0 hrs, 3 min, 45 sec │ │ cycles done : 12 │
│ last new find : none yet (odd, check syntax!) │ │ corpus count : 3 │
│ last saved crash : none seen yet │ │ saved crashes : 0 │
│ last saved hang : none seen yet │ │ saved hangs : 0 │
└──────────┴──────────┘
┌──────────┴──────────┐
┌ cycle progress ────┐ ┌ map coverage ────┐
│ now processing : 2.60 (66.7%) │ │ map density : 0.00% / 0.00% │
│ runs timed out : 0 (0.00%) │ │ count coverage : 4.50 bits/tuple │
└──────────┴──────────┘ ┌ findings in depth ────┐
┌ stage progress ────┐ │ favored items : 1 (33.33%) │
│ now trying : havoc │ │ new edges on : 1 (33.33%) │
│ stage execs : 1320/1766 (74.75%) │ │ total crashes : 0 (0 saved) │
│ total execs : 106k │ │ total tmouts : 0 (0 saved) │
│ exec speed : 452.1/sec │ └──────────┴──────────┘
└──────────┴──────────┘ ┌ item geometry ────┐
┌ fuzzing strategy yields ────┐ │ levels : 1 │
│ bit flips : disabled (default, enable with -D) │ │ pending : 0 │
│ byte flips : disabled (default, enable with -D) │ │ pend fav : 0 │
│ arithmetics : disabled (default, enable with -D) │ │ own finds : 0 │
│ known ints : disabled (default, enable with -D) │ │ imported : 0 │
│ dictionary : n/a │ │ stability : 50.00% │
│ havoc/splice : 0/104k, 0/0 │ │ │
│ py/custom/rq : unused, unused, unused, unused │ │ │
│ trim/eff : 20.00%/1, disabled │ │ │
└──────────┴──────────┘ └──────────┴──────────┘
[cpu000: 33%]

```

- 0 crash and 0 hang were detected by AFL with sanitizer *signed-integer-overflow*

```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/vulnerable/integer
File Edit View Search Terminal Help

american fuzzy lop ++3.15a {default} (./AFL2) [fast]
┌──────────┴──────────┐
┌ process timing ────┐ ┌ overall results ────┐
│ run time : 0 days, 0 hrs, 5 min, 18 sec │ │ cycles done : 64 │
│ last new find : none yet (odd, check syntax!) │ │ corpus count : 1 │
│ last saved crash : none seen yet │ │ saved crashes : 0 │
│ last saved hang : none seen yet │ │ saved hangs : 0 │
└──────────┴──────────┘
┌──────────┴──────────┐
┌ cycle progress ────┐ ┌ map coverage ────┐
│ now processing : 0.194 (0.0%) │ │ map density : 0.00% / 0.00% │
│ runs timed out : 0 (0.00%) │ │ count coverage : 4.00 bits/tuple │
└──────────┴──────────┘ ┌ findings in depth ────┐
┌ stage progress ────┐ │ favored items : 1 (100.00%) │
│ now trying : havoc │ │ new edges on : 1 (100.00%) │
│ stage execs : 270/587 (46.00%) │ │ total crashes : 0 (0 saved) │
│ total execs : 113k │ │ total tmouts : 0 (0 saved) │
│ exec speed : 483.6/sec │ └──────────┴──────────┘
└──────────┴──────────┘ ┌ item geometry ────┐
┌ fuzzing strategy yields ────┐ │ levels : 1 │
│ bit flips : disabled (default, enable with -D) │ │ pending : 0 │
│ byte flips : disabled (default, enable with -D) │ │ pend fav : 0 │
│ arithmetics : disabled (default, enable with -D) │ │ own finds : 0 │
│ known ints : disabled (default, enable with -D) │ │ imported : 0 │
│ dictionary : n/a │ │ stability : 57.14% │
│ havoc/splice : 0/113k, 0/0 │ │ │
│ py/custom/rq : unused, unused, unused, unused │ │ │
│ trim/eff : 0.00%/1, disabled │ │ │
└──────────┴──────────┘ └──────────┴──────────┘
[cpu001: 50%]

```

- 0 crash and 0 hang were detected by AFL with sanitizer *address*

```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/vulnerable/address
File Edit View Search Terminal Help

american fuzzy lop ++3.15a {default} (./AFL2) [fast]
- process timing
  run time : 0 days, 0 hrs, 5 min, 58 sec
  last new find : none yet (odd, check syntax!)
  last saved crash : none seen yet
  last saved hang : none seen yet
- cycle progress
  now processing : 0.235 (0.0%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : havoc
  stage execs : 480/587 (81.77%)
  total execs : 137k
  exec speed : 347.3/sec
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 0/137k, 0/0
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 0.00%/1, disabled
- map coverage
  map density : 0.00% / 0.00%
  count coverage : 4.50 bits/tuple
- findings in depth
  favored items : 1 (100.00%)
  new edges on : 1 (100.00%)
  total crashes : 0 (0 saved)
  total tmouts : 0 (0 saved)
- item geometry
  levels : 1
  pending : 0
  pend fav : 0
  own finds : 0
  imported : 0
  stability : 50.00%
[cpu000: 50%]

```

- 0 crash and 0 hang were detected by AFL with sanitizer *undefined*

```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/vulnerable/undefined
File Edit View Search Terminal Help

american fuzzy lop ++3.15a {default} (./AFL2) [fast]
- process timing
  run time : 0 days, 0 hrs, 4 min, 20 sec
  last new find : none yet (odd, check syntax!)
  last saved crash : none seen yet
  last saved hang : none seen yet
- cycle progress
  now processing : 1.53 (33.3%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : havoc
  stage execs : 712/2352 (30.27%)
  total execs : 123k
  exec speed : 450.2/sec
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 0/123k, 0/0
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 0.00%/3, disabled
- map coverage
  map density : 0.00% / 0.00%
  count coverage : 4.50 bits/tuple
- findings in depth
  favored items : 1 (33.33%)
  new edges on : 1 (33.33%)
  total crashes : 0 (0 saved)
  total tmouts : 0 (0 saved)
- item geometry
  levels : 1
  pending : 0
  pend fav : 0
  own finds : 0
  imported : 0
  stability : 50.00%
[cpu000: 33%]

```

ii) Try fixing a few bugs(s) in the program, and running AFL on it again. Do you discover any new bugs or paths?

We use the bug-fixed program in Task 4 to run AFL again.

- 0 crash and 0 hang were detected by AFL without sanitizer

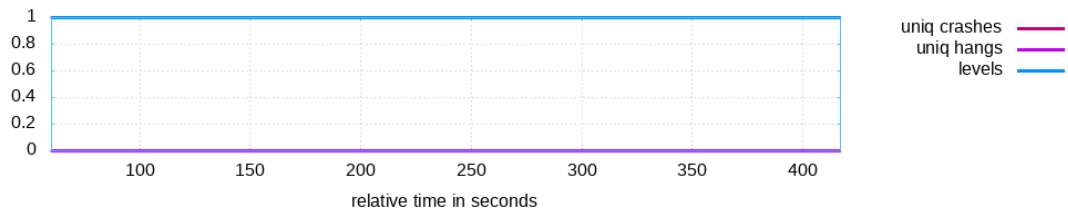
```

cpen@cpenUBC-VirtualBox: ~/AFLplusplus/vulnerable/modifiedBugs
File Edit View Search Terminal Help

american fuzzy lop ++3.15a {default} (./AFL2) [fast]
- process timing
  run time : 0 days, 0 hrs, 5 min, 32 sec
  last new find : none yet (odd, check syntax!)
  last saved crash : none seen yet
  last saved hang : none seen yet
- cycle progress
  now processing : 0.252 (0.0%)
  runs timed out : 0 (0.00%)
- stage progress
  now trying : havoc
  stage execs : 208/587 (35.43%)
  total execs : 147k
  exec speed : 436.2/sec
- fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 0/147k, 0/0
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 0.00%/1, disabled
- map coverage
  map density : 0.00% / 0.00%
  count coverage : 4.50 bits/tuple
- findings in depth
  favored items : 1 (100.00%)
  new edges on : 1 (100.00%)
  total crashes : 0 (0 saved)
  total tmouts : 0 (0 saved)
- item geometry
  levels : 1
  pending : 0
  pend fav : 0
  own finds : 0
  imported : 0
  stability : 50.00%
[cpu000: 33%]

```

- 0 crash and 0 hang were detected by AFL with sanitizer *signed-integer-overflow*



iv) Compare the results using KLEE and AFL on *Vulnerable.c*. Which tool is preferable for bug detection? What are the advantages and drawbacks of using each tool?

- KLEE is preferable for bug detection on *Vulnerable.c* because no bugs detected by AFL in our experiment.
- KLEE:
 - Pros: Because KLEE is a concolic execution tool, it can trigger the bug with precise input values, which help developers quickly identifying and fixing issues.
 - Cons: KLEE's focus on path exploration but it may not be able to handle system calls, and external dependencies effectively like AFL.
- AFL:
 - Pros: AFL can handle system calls, and external dependencies more effectively than KLEE.
 - Cons: AFL heavily relies on random mutations of inputs, which may lead to limited path coverage. Also, the large number of test inputs will lead to longer running time.