Lab 7

Software Testing 2023

2023/04/27

#whoami

- Software Quality Lab @ EC547
- TA
  a. 蔡惠喬
     - hctsai.cs10@nycu.edu.tw
  b. 陳舜寧
     - xdev11.cs11@nycu.edu.tw

GitHub Repo

- <student_id>-ST-2023
- Add collaborators
  - XDEv11, chameleon10712, skhuang

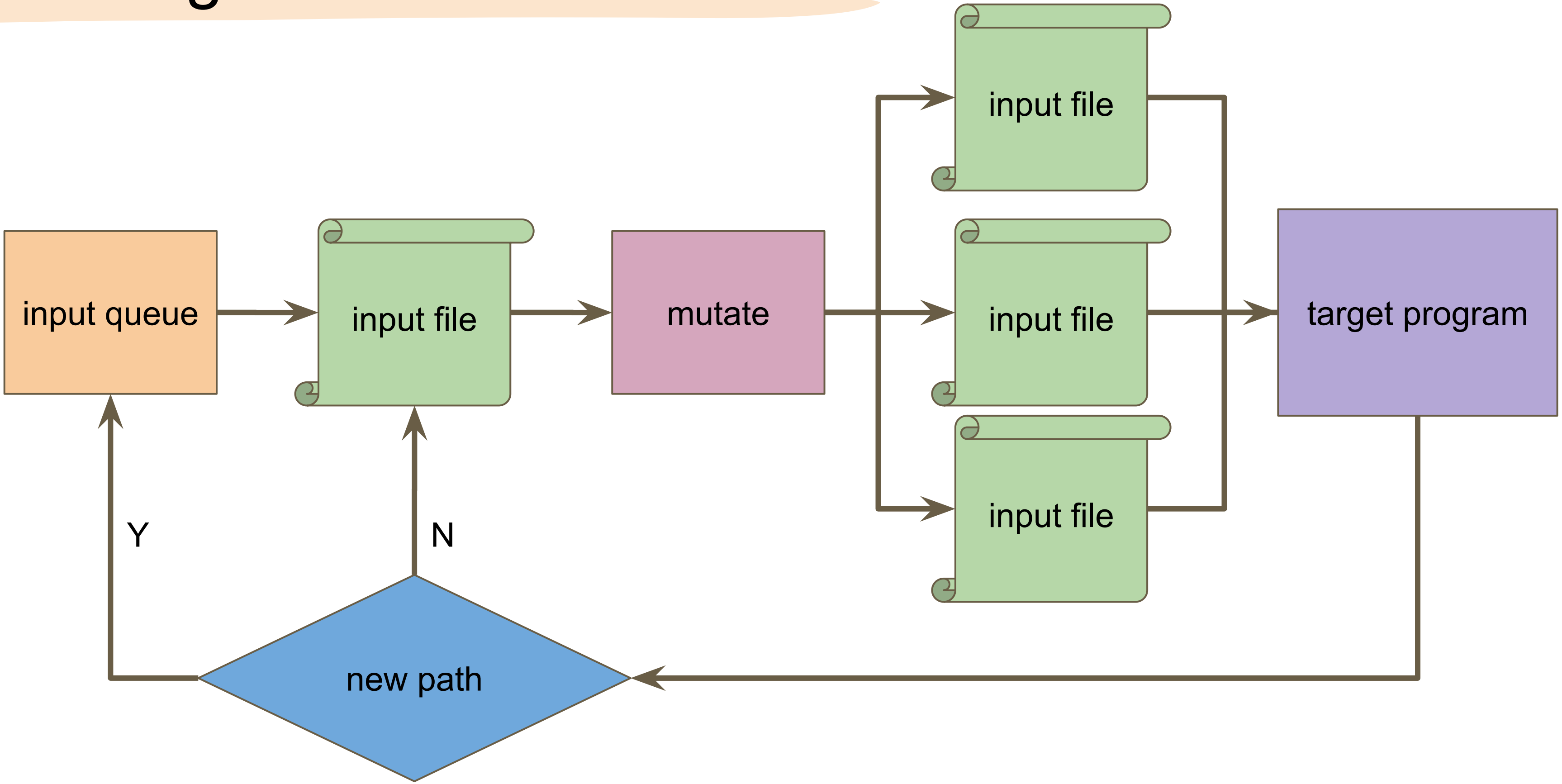
Fuzz Testing

Traditional testing procedures

- Unit Test
  - Since it is **manual**, it is **difficult** to consider all.
    - Is there any problem with function combination?
    - Are the inputs that are not in the specification segregated?
    - Are some inputs related to internal memory config well handled?

- Can the whole program be tested automatically?

Fuzzing Architecture

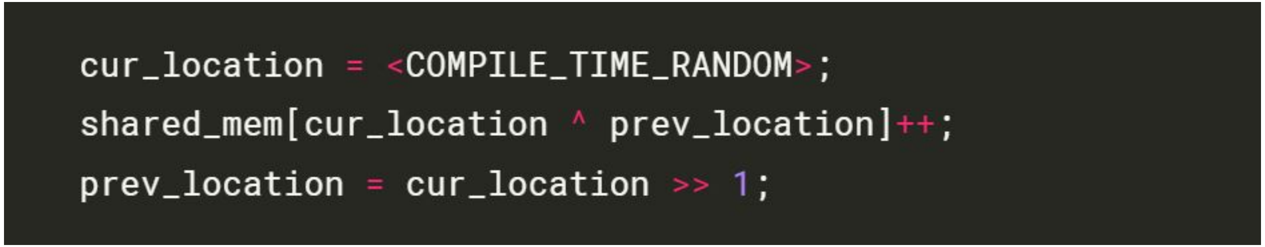


Code Coverage

- It is **difficult** to know if the input is good after mutate.
    - Currently, the most common method is based on **code coverage**.
        - Hope to cover the **uncovered** Basic Blocks.
        - Hope to cover the **more** Basic Blocks.

Instrumentation

- How to get the execution status of the program quickly?
  - Have Source Code
    - Instrumentation through compilation tools such as gcc, clang, LLVM, etc.
      - Add specific code in front of each basic block.



```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

  - No Source Code
    - Binary direct rewriting
    - Simulator (Qemu, Unicorn, Qiling)

Mutate

- Generate input by mutating existing files
  - **bitflip x/y :** bit flip
  - **arithmetic x/y :** adding or subtracting an integer
  - **interest x/y :** replace the bits with the data of interest
    - ex : INT_MAX , 0
  - **dictionary :** the token provided by the source user, and the token generated by automatic detection
  - **havoc :** combination of multiple mutation methods
  - **splice :** 2 seeds are spliced and havoc is performed

AFL

- Introduced by Google.
- The pioneer of coverage-guided.
- However, there have been no major updates since 2017.
  - So …

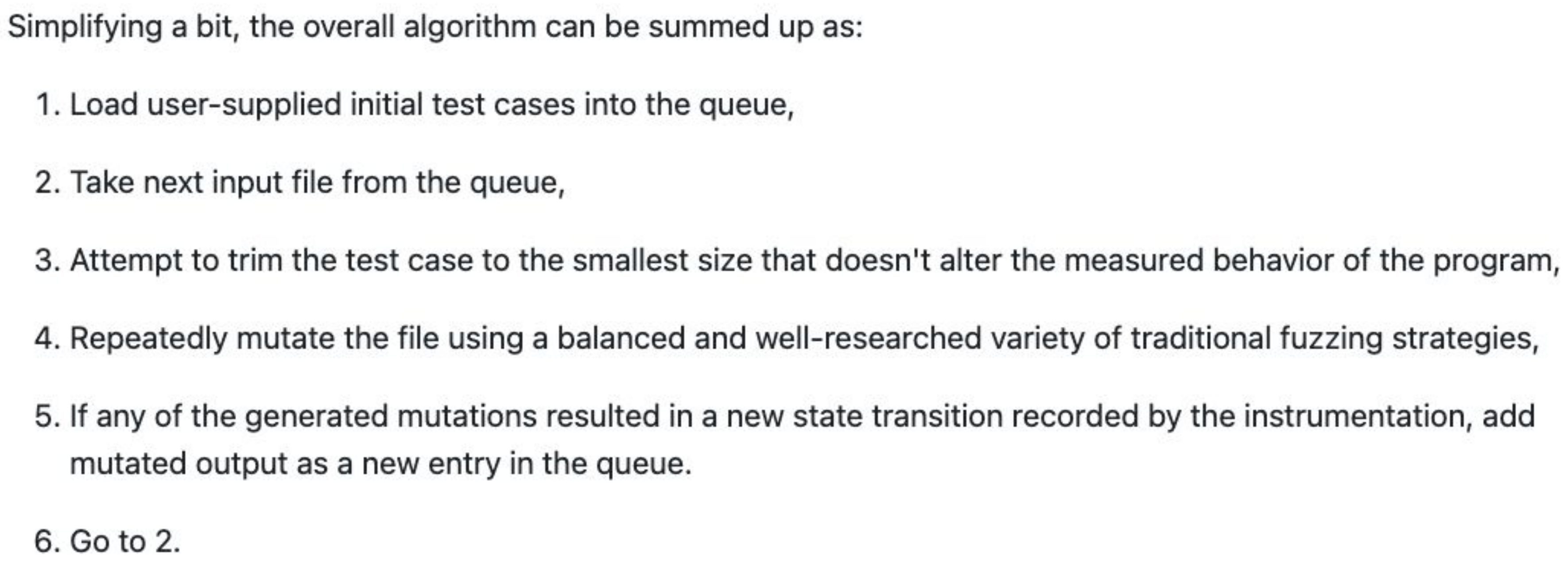
AFL++

- Extensive fuzz testing community.

- Collection of quality papers and improvements.

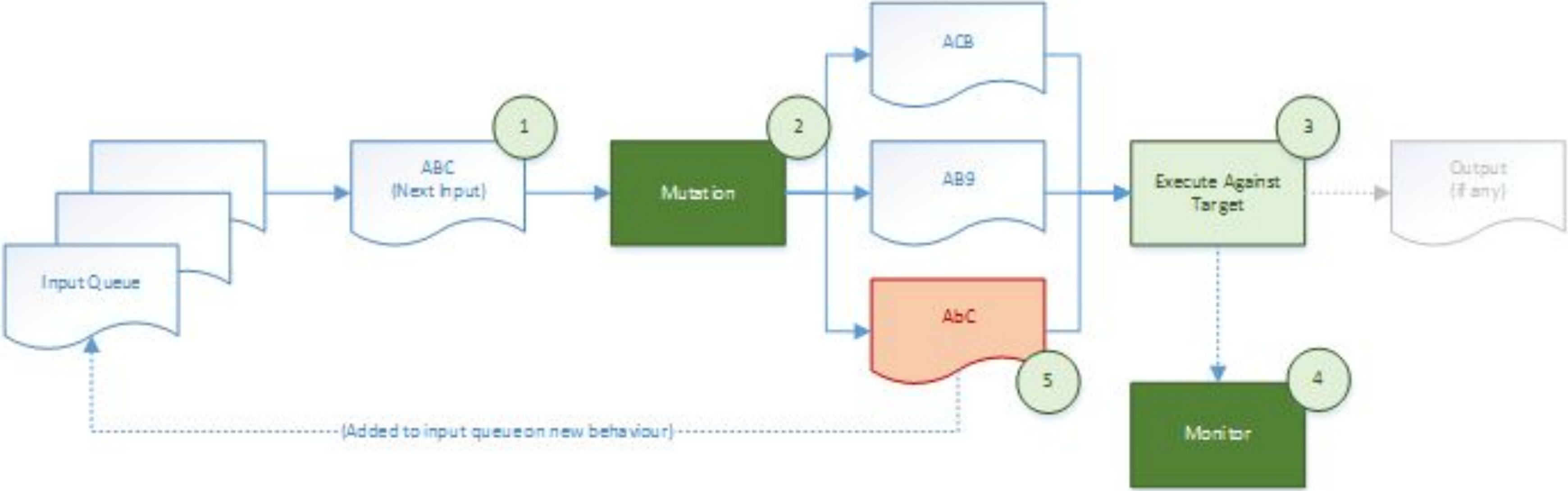
- Continuous updates and integration of new fuzzy testing techniques.
  - example : using deep learning, new mutation techniques, etc.

AFL



Simplifying a bit, the overall algorithm can be summed up as:

1. Load user-supplied initial test cases into the queue,
2. Take next input file from the queue,
3. Attempt to trim the test case to the smallest size that doesn't alter the measured behavior of the program,
4. Repeatedly mutate the file using a balanced and well-researched variety of traditional fuzzing strategies,
5. If any of the generated mutations resulted in a new state transition recorded by the instrumentation, add mutated output as a new entry in the queue.
6. Go to 2.


Input Queue
1
ABC
(Next Input)
2
Mutation
ACB
AB9
AbC
5
3
Execute Against Target
Output
(if any)
4
Monitor
(Added to input queue on new behaviour)

Installation on Linux

```
$ git clone https://github.com/google/AFL.git

$ cd AFL

$ make

$ sudo make install
```

Reference: doc/INSTALL

Example

libxml2

```
$ git clone https://gitlab.gnome.org/GNOME/libxml2.git
$ cd libxml2
$ ./autogen.sh
$ export CC=~/AFL/afl-gcc
$ export CXX=~/AFL/afl-g++
$ export AFL_USE_ASAN=1
$ ./configure --enable-shared=no
$ make
```

libxml2

```
$ mkdir -p  ~/fuzz/in

$ cp xmllint  ~/fuzz/libxml2

$ cp test/*.xml   ~/fuzz/in/
```

Fuzzing with AFL

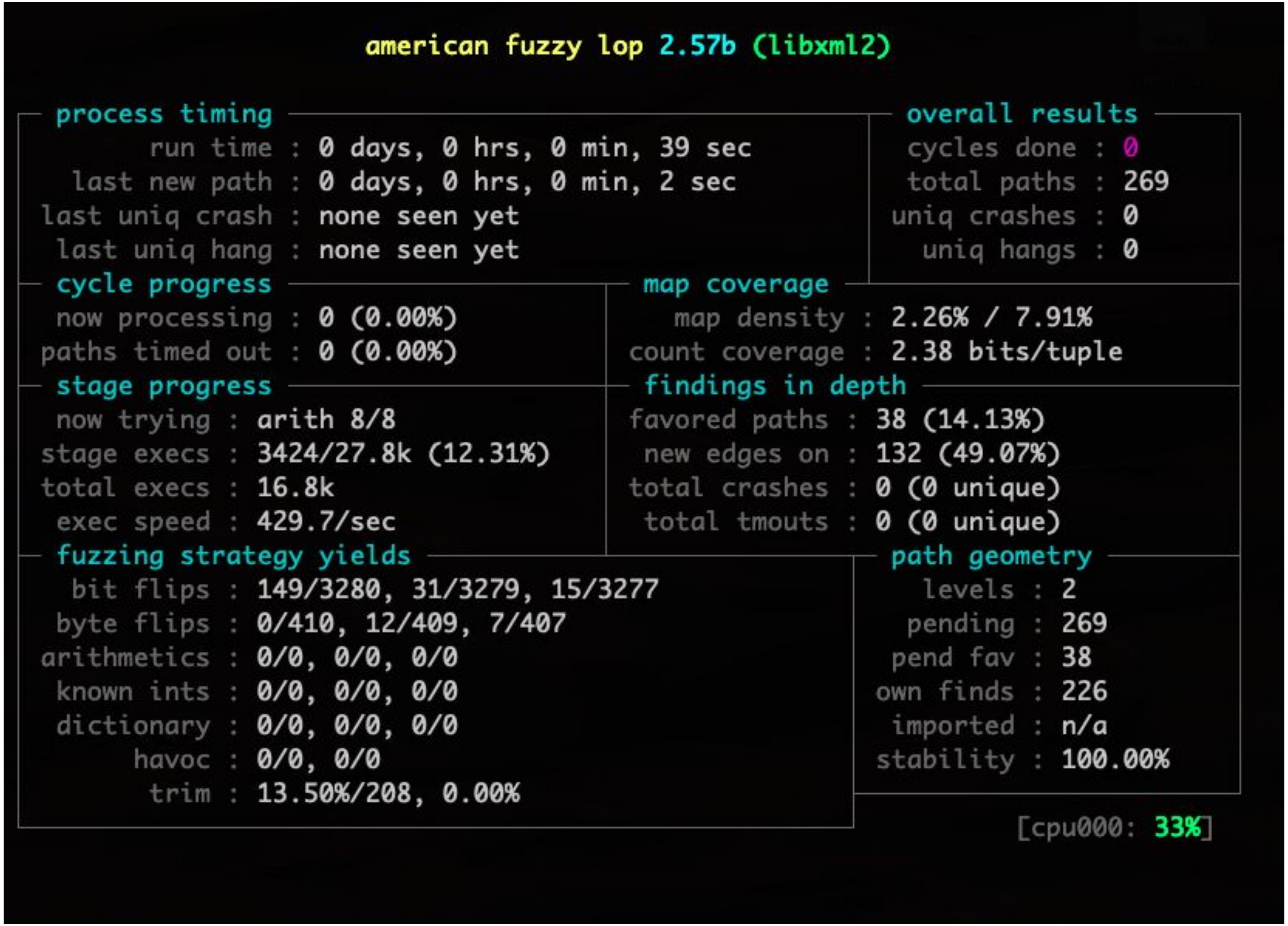
```
$ cd ~/fuzz

$ ~/AFL/afl-fuzz -i in/ -o out/ -m none -- ./libxml2 @@
```

Fuzzing

```
$ afl-fuzz -i in/ -o out/ -b 10 -m none -- ./target [argv1] @@ [argv2]
```

- -i dir : seed dir
- -o dir : output dir
- -b CPU_ID : bind the fuzzing process to the specified CPU core
- -m megs: memory limit for child process
- @@ : the location of the input (if NO -> stdin)



```
                    american fuzzy lop 2.57b (libxml2)

┌─ process timing ─────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 39 sec       │  cycles done : 0     │
│   last new path : 0 days, 0 hrs, 0 min, 2 sec        │  total paths : 269   │
│ last uniq crash : none seen yet                      │ uniq crashes : 0     │
│  last uniq hang : none seen yet                      │   uniq hangs : 0     │
├─ cycle progress ─────────────────┬─ map coverage ────┴──────────────────────┤
│  now processing : 0 (0.00%)      │    map density : 2.26% / 7.91%          │
│ paths timed out : 0 (0.00%)      │ count coverage : 2.38 bits/tuple        │
├─ stage progress ─────────────────┼─ findings in depth ──────────────────────┤
│  now trying : arith 8/8          │ favored paths : 38 (14.13%)             │
│ stage execs : 3424/27.8k (12.31%)│  new edges on : 132 (49.07%)            │
│ total execs : 16.8k              │ total crashes : 0 (0 unique)            │
│  exec speed : 429.7/sec          │  total tmouts : 0 (0 unique)            │
├─ fuzzing strategy yields ────────┴───────────────┬─ path geometry ──────────┤
│   bit flips : 149/3280, 31/3279, 15/3277         │    levels : 2            │
│  byte flips : 0/410, 12/409, 7/407               │   pending : 269          │
│ arithmetics : 0/0, 0/0, 0/0                      │  pend fav : 38           │
│  known ints : 0/0, 0/0, 0/0                      │ own finds : 226          │
│  dictionary : 0/0, 0/0, 0/0                      │  imported : n/a          │
│       havoc : 0/0, 0/0                           │ stability : 100.00%      │
│        trim : 13.50%/208, 0.00%                  ├──────────────────────────┘
└──────────────────────────────────────────────────┘          [cpu000: 33%]
```

Fuzzing - Result

- **queue/** - test cases for every distinctive execution path, plus all the starting files given by the user.
- **crashes/** - unique test cases that cause the tested program to receive a fatal signal (e.g., SIGSEGV, SIGILL, SIGABRT).
- **hangs/** - unique test cases that cause the tested program to time out. The default time limit before something is classified as a hang is the larger of 1 second and the value of the -t parameter.

Lab 7

Lab 7

- We provide a small program that converts bmp from color to grayscale.
  - Use **AFL** to find the file that can trigger the vulnerability.
  - Use **test.bmp** as initial seed.

- Please write a report named README.md/rst placed in lab07 dir in your github repo.
- The report shall contain the following information:
  - PoC: the file that can trigger the vulnerability
  - The commands (steps) that you used in this lab
  - Screenshot of AFL running (with triggered crash)
  - Screenshot of crash detail (with ASAN error report)

Lab 7

- Download target source code from: NYCU-Software-Testing-2023/Lab07

Lab 7

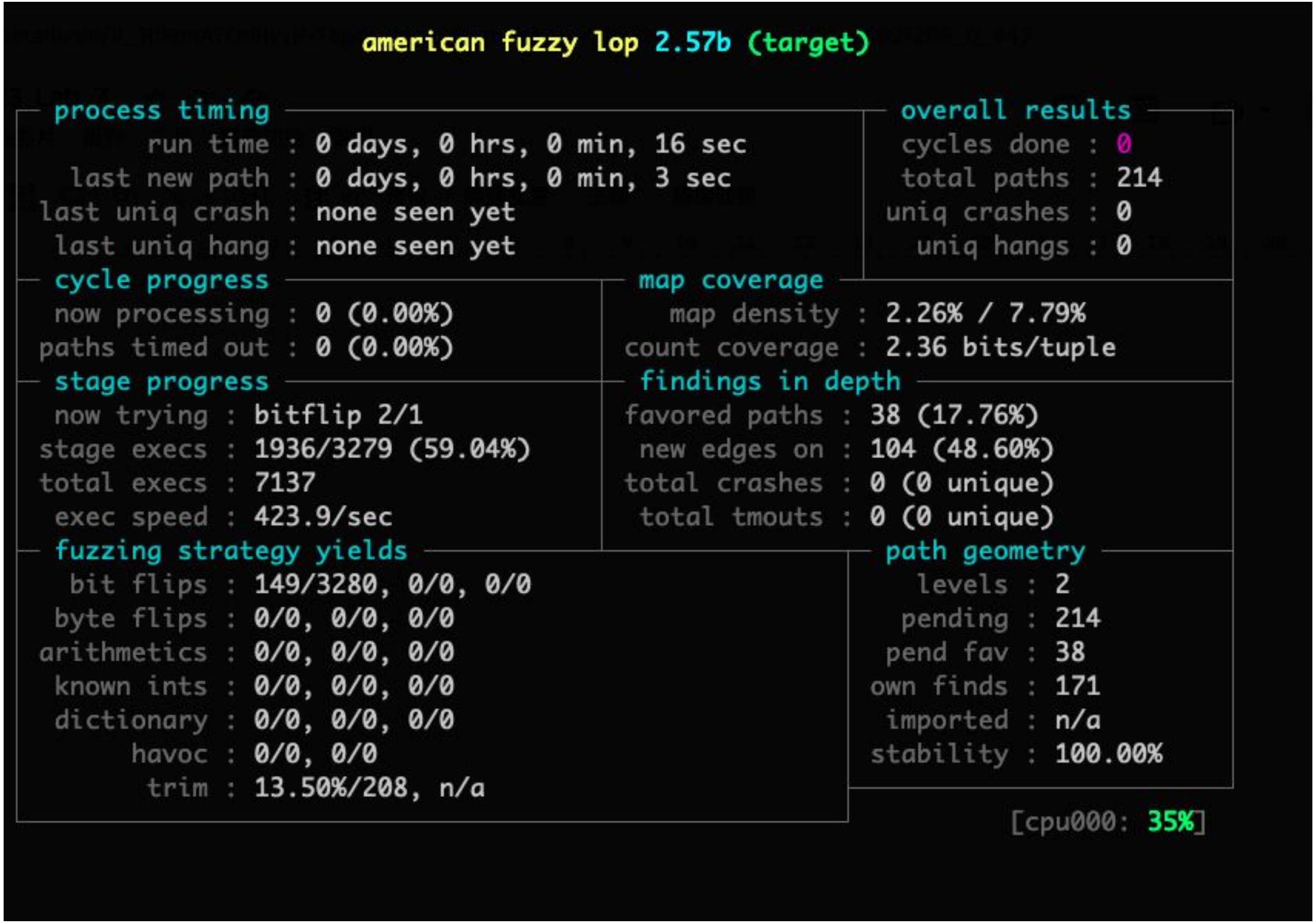
Build & fuzz with AFL

```
$ cd Lab07
$ export CC=~/AFL/afl-gcc
$ export AFL_USE_ASAN=1
$ make
$ mkdir in
$ cp test.bmp in/
$ ~/AFL/afl-fuzz -i in -o out -m none -- ./bmpgrayscale @@ a.bmp
```

Lab 7

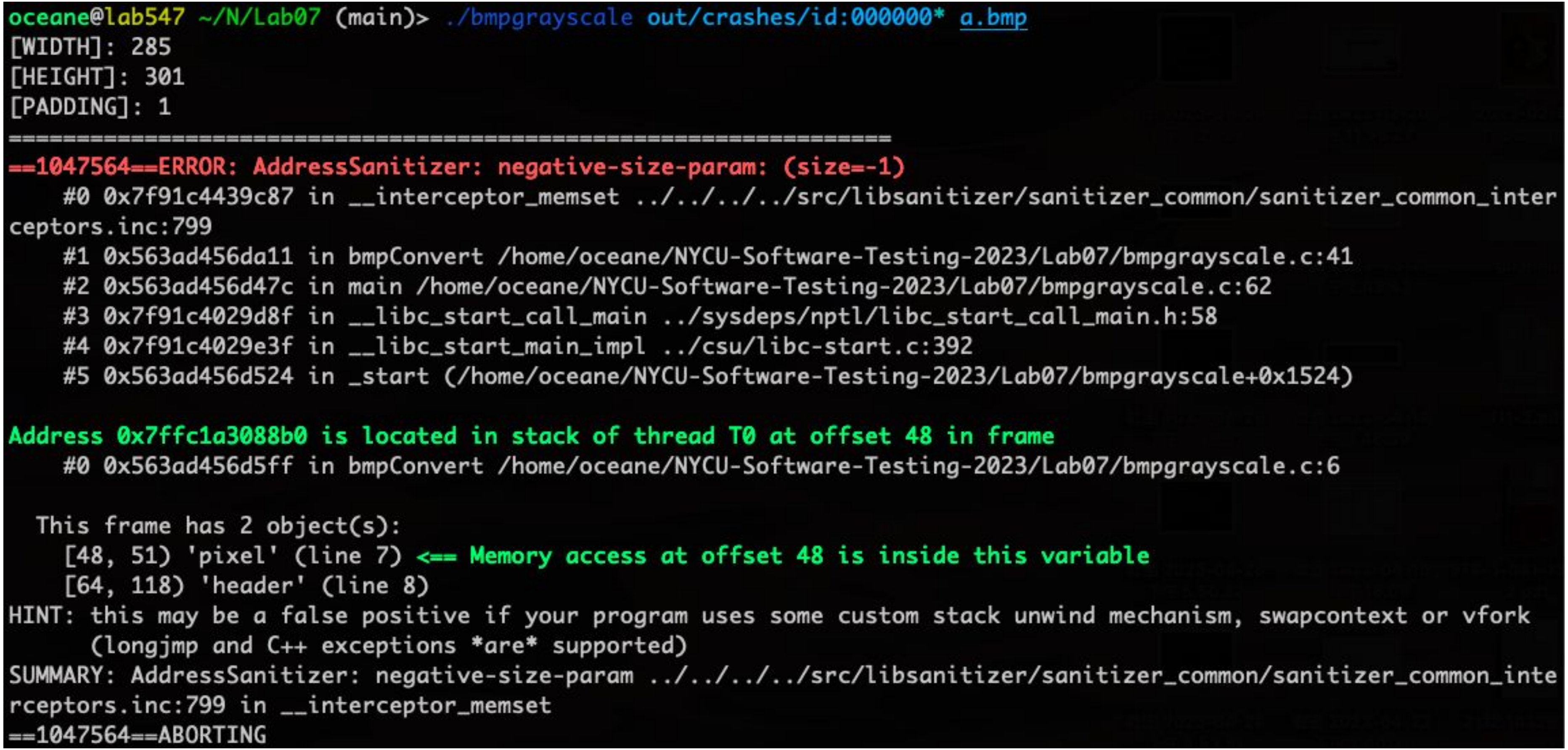
```
$ ./bmpgrayscale out/crashes/id:000000* a.bmp
```

Example Output



```
                    american fuzzy lop 2.57b (target)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 0 min, 16 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 3 sec        │  total paths : 214    │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────┬─ map coverage ─────┴───────────────────────┤
│  now processing : 0 (0.00%)     │    map density : 2.26% / 7.79%            │
│ paths timed out : 0 (0.00%)     │ count coverage : 2.36 bits/tuple          │
├─ stage progress ────────────────┼─ findings in depth ────────────────────────┤
│  now trying : bitflip 2/1       │ favored paths : 38 (17.76%)               │
│ stage execs : 1936/3279 (59.04%)│  new edges on : 104 (48.60%)              │
│ total execs : 7137              │ total crashes : 0 (0 unique)              │
│  exec speed : 423.9/sec         │  total tmouts : 0 (0 unique)              │
├─ fuzzing strategy yields ───────┴───────────────┬─ path geometry ────────────┤
│   bit flips : 149/3280, 0/0, 0/0                │    levels : 2              │
│  byte flips : 0/0, 0/0, 0/0                     │   pending : 214            │
│ arithmetics : 0/0, 0/0, 0/0                     │  pend fav : 38             │
│  known ints : 0/0, 0/0, 0/0                     │ own finds : 171            │
│  dictionary : 0/0, 0/0, 0/0                     │  imported : n/a            │
│       havoc : 0/0, 0/0                          │ stability : 100.00%        │
│        trim : 13.50%/208, n/a                   ├────────────────────────────┘
└─────────────────────────────────────────────────┘             [cpu000: 35%]
```



```
oceane@lab547 ~/N/Lab07 (main)> ./bmpgrayscale out/crashes/id:000000* a.bmp
[WIDTH]: 285
[HEIGHT]: 301
[PADDING]: 1
=================================================================
==1047564==ERROR: AddressSanitizer: negative-size-param: (size=-1)
    #0 0x7f91c4439c87 in __interceptor_memset ../../../../src/libsanitizer/sanitizer_common/sanitizer_common_inter
ceptors.inc:799
    #1 0x563ad456da11 in bmpConvert /home/oceane/NYCU-Software-Testing-2023/Lab07/bmpgrayscale.c:41
    #2 0x563ad456d47c in main /home/oceane/NYCU-Software-Testing-2023/Lab07/bmpgrayscale.c:62
    #3 0x7f91c4029d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
    #4 0x7f91c4029e3f in __libc_start_main_impl ../csu/libc-start.c:392
    #5 0x563ad456d524 in _start (/home/oceane/NYCU-Software-Testing-2023/Lab07/bmpgrayscale+0x1524)

Address 0x7ffc1a3088b0 is located in stack of thread T0 at offset 48 in frame
    #0 0x563ad456d5ff in bmpConvert /home/oceane/NYCU-Software-Testing-2023/Lab07/bmpgrayscale.c:6

  This frame has 2 object(s):
    [48, 51) 'pixel' (line 7) <== Memory access at offset 48 is inside this variable
    [64, 118) 'header' (line 8)
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: negative-size-param ../../../../src/libsanitizer/sanitizer_common/sanitizer_common_inte
rceptors.inc:799 in __interceptor_memset
==1047564==ABORTING
```

Submission

Submission

- Please submit your Github repo <student_id>-ST-2023 (1) commit URL to E3
- Please submit your URL as link
- commit URL
  - refer to Lab 1 submission

Reference

Reference

- https://github.com/google/AFL
- https://afl-1.readthedocs.io/en/latest/
- https://github.com/AFLplusplus/AFLplusplus
- https://aflplus.plus/docs/technical_details/
- https://aflplus.plus/docs/tutorials/libxml2_tutorial/