Lab 7: Fuzz Testing

Software Testing 2022

2022/04/21

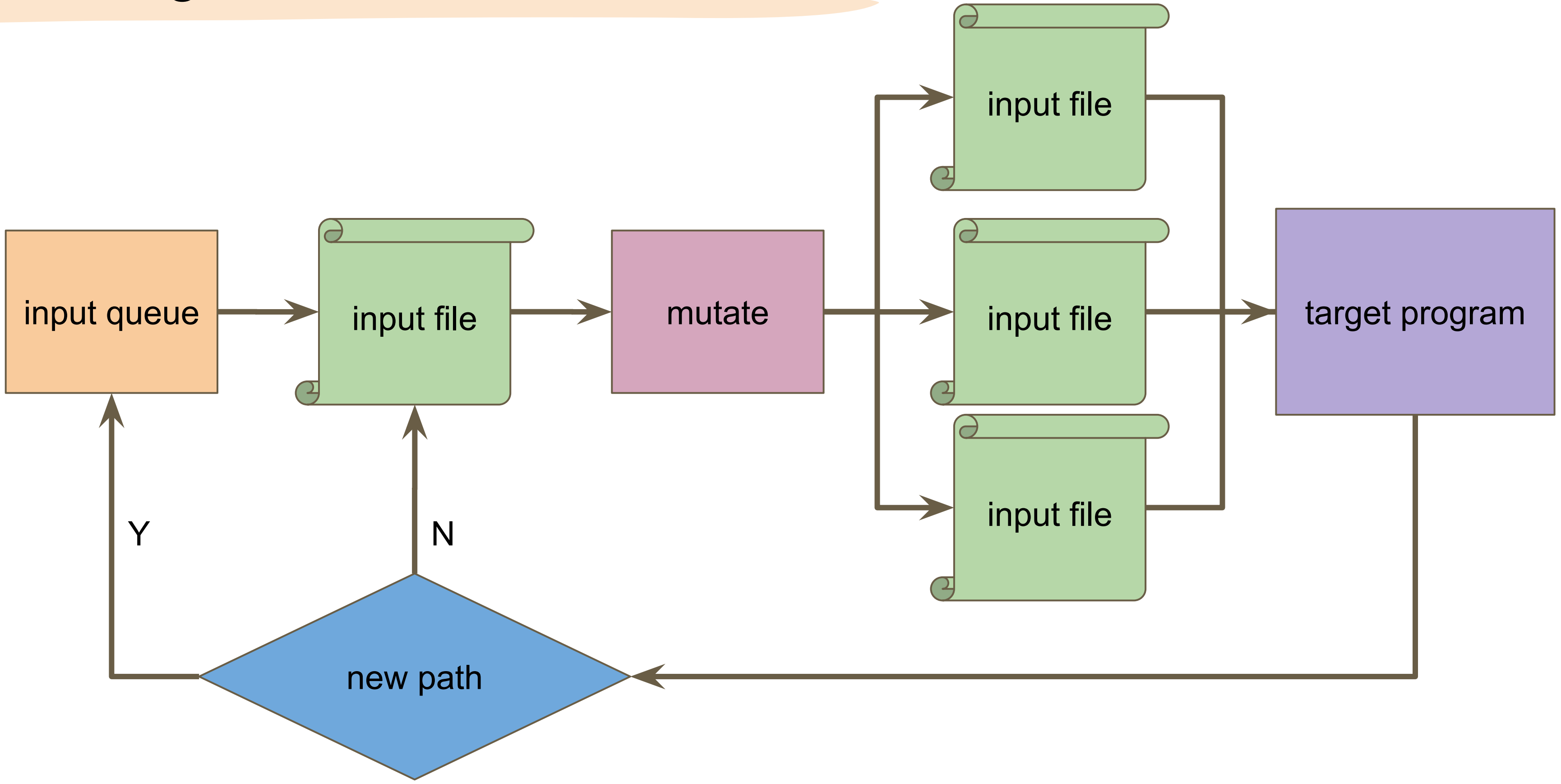
Introduction

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

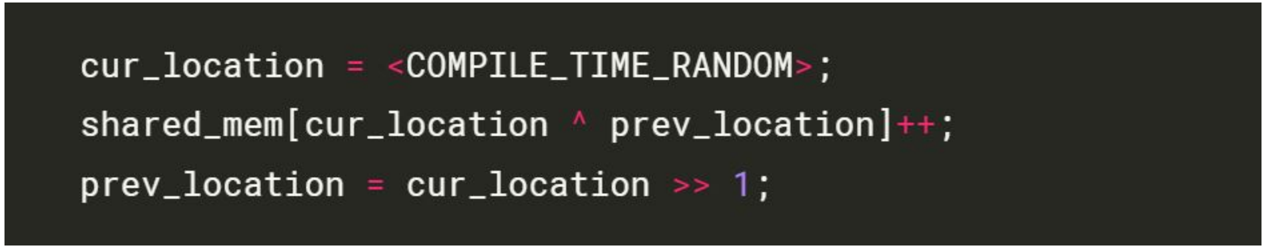


Code Coverage

- 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.

Build & Install

- https://github.com/AFLplusplus/AFLplusplus/blob/stable/docs/INSTALL.md

Result

- Instrument's Compiler:
  - afl-cc / afl-c++
  - afl-gcc / afl-g++
  - afl-gcc-fast / afl-g++-fast
  - afl-clang / afl-clang++
  - afl-clang-fast / afl-clang-fast++
  - afl-clang-lto / afl-clang–lto++

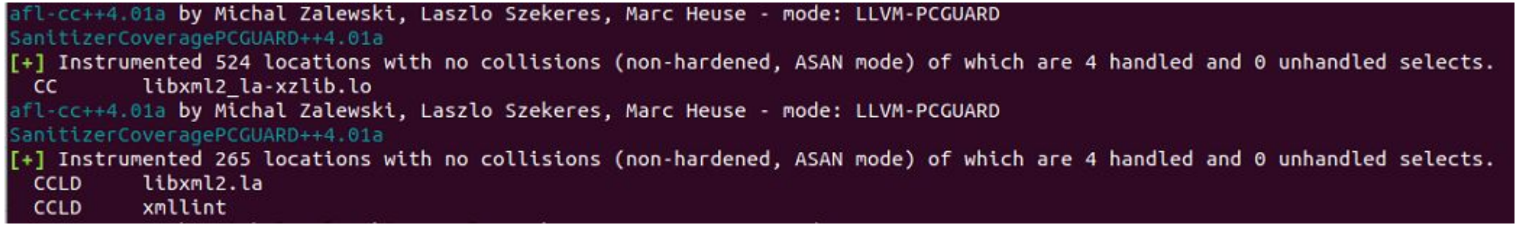
- afl-fuzz

- afl-showmap afl-cmin afl-tmin ...

Example

How to Fuzzing? (libxml2)

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

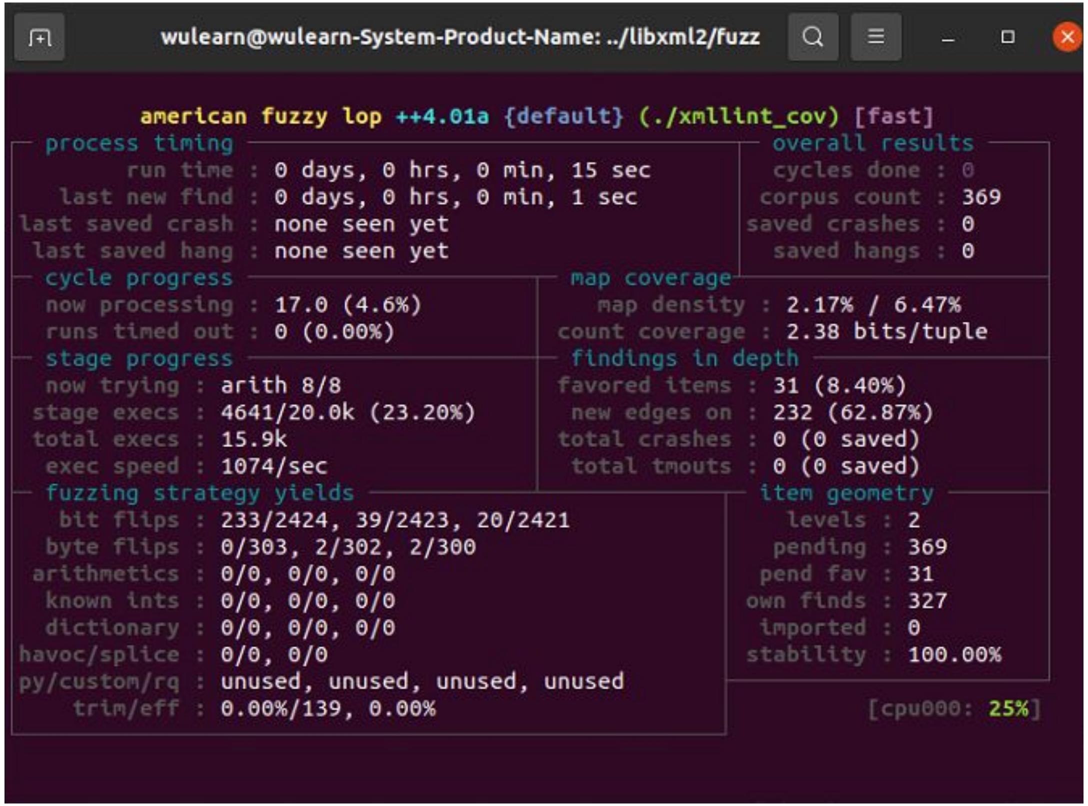


```
afl-cc++4.01a by Michal Zalewski, Laszlo Szekeres, Marc Heuse - mode: LLVM-PCGUARD
SanitizerCoveragePCGUARD++4.01a
[+] Instrumented 524 locations with no collisions (non-hardened, ASAN mode) of which are 4 handled and 0 unhandled selects.
  CC       libxml2_la-xzlib.lo
afl-cc++4.01a by Michal Zalewski, Laszlo Szekeres, Marc Heuse - mode: LLVM-PCGUARD
SanitizerCoveragePCGUARD++4.01a
[+] Instrumented 265 locations with no collisions (non-hardened, ASAN mode) of which are 4 handled and 0 unhandled selects.
  CCLD     libxml2.la
  CCLD     xmllint
```

How to Fuzzing? (libxml2)

```
$ cp xmllint fuzz/xmllint_cov
$ mkdir fuzz/in
$ cp test/*.xml fuzz/in/
$ cd fuzz
$ ~/AFLplusplus/afl-fuzz -i in/ -o out/ -m none -D -- ./xmllint_cov @@
```

Fuzzing - Screenshot



wulearn@wulearn-System-Product-Name: ../libxml2/fuzz

```
         american fuzzy lop ++4.01a {default} (./xmllint_cov) [fast]
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 15 sec      │  cycles done : 0     │
│   last new find : 0 days, 0 hrs, 0 min, 1 sec       │ corpus count : 369   │
│last saved crash : none seen yet                     │saved crashes : 0     │
│ last saved hang : none seen yet                     │  saved hangs : 0     │
├─ cycle progress ───────────────────┬─ map coverage─┴──────────────────────┤
│  now processing : 17.0 (4.6%)      │    map density : 2.17% / 6.47%      │
│  runs timed out : 0 (0.00%)        │ count coverage : 2.38 bits/tuple    │
├─ stage progress ───────────────────┼─ findings in depth ─────────────────┤
│  now trying : arith 8/8            │ favored items : 31 (8.40%)          │
│ stage execs : 4641/20.0k (23.20%)  │  new edges on : 232 (62.87%)        │
│ total execs : 15.9k                │ total crashes : 0 (0 saved)         │
│  exec speed : 1074/sec             │  total tmouts : 0 (0 saved)         │
├─ fuzzing strategy yields ──────────┴─────────────┬─ item geometry ───────┤
│   bit flips : 233/2424, 39/2423, 20/2421         │    levels : 2         │
│  byte flips : 0/303, 2/302, 2/300                │   pending : 369       │
│ arithmetics : 0/0, 0/0, 0/0                      │  pend fav : 31        │
│  known ints : 0/0, 0/0, 0/0                      │ own finds : 327       │
│  dictionary : 0/0, 0/0, 0/0                      │  imported : 0         │
│havoc/splice : 0/0, 0/0                           │ stability : 100.00%   │
│py/custom/rq : unused, unused, unused, unused     ├───────────────────────┘
│    trim/eff : 0.00%/139, 0.00%                   │          [cpu000: 25%]
└──────────────────────────────────────────────────┘
```

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)

Fuzzing - Result

- In …/out_dir/default
  - crashes
  - hangs
  - queue

Lab

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 init seed.

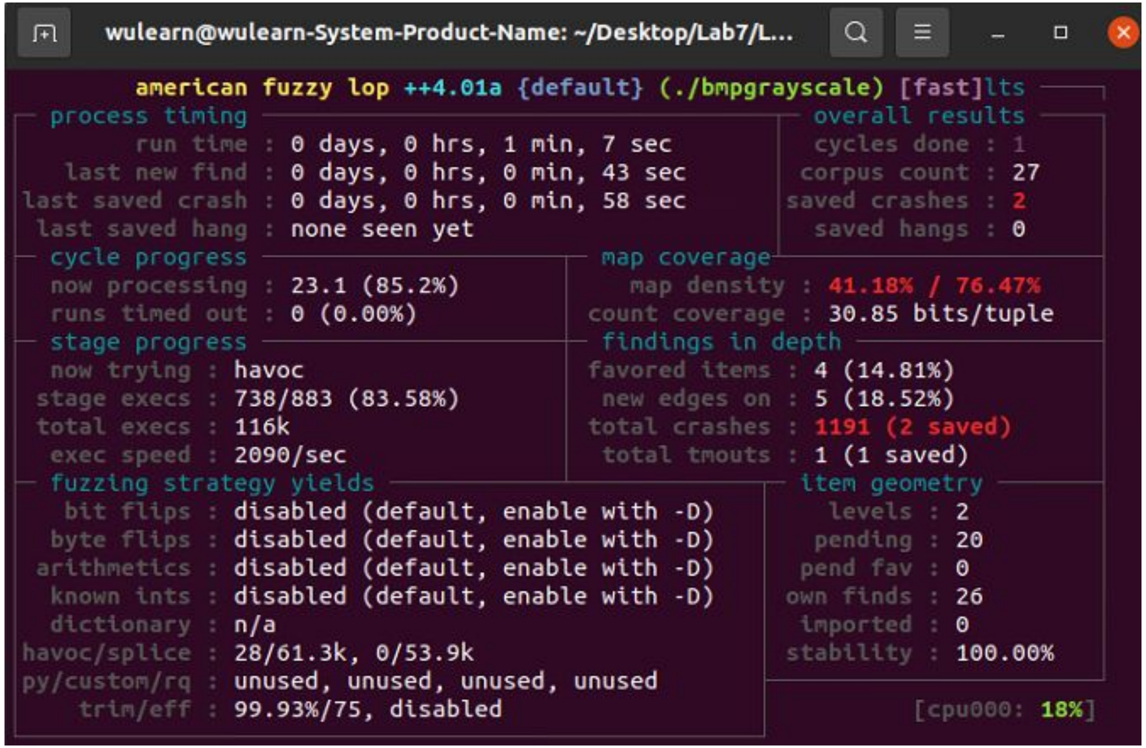
- Deliverables shall include the following:
    - PoC: the file that can trigger the vulnerability
    - Screenshot of AFL++ running (with triggered crash): STUDENT_ID.png

- **Do not compress the files and plagiarism!**

Lab 7

```
$ Build & Install AFL++
$ git clone https://github.com/a4865g/NYCU-Software-Testing-2022.git
$ cd NYCU-Software-Testing-2022/Lab_7
$ export CC=~/AFLplusplus/afl-cc
$ export AFL_USE_ASAN=1
$ make
$ mkdir in
$ cp test.bmp in/
$ ~/AFLplusplus/afl-fuzz -i in -o out -m none -- ./bmpgrayscale @@ a.bmp
```

Lab 7

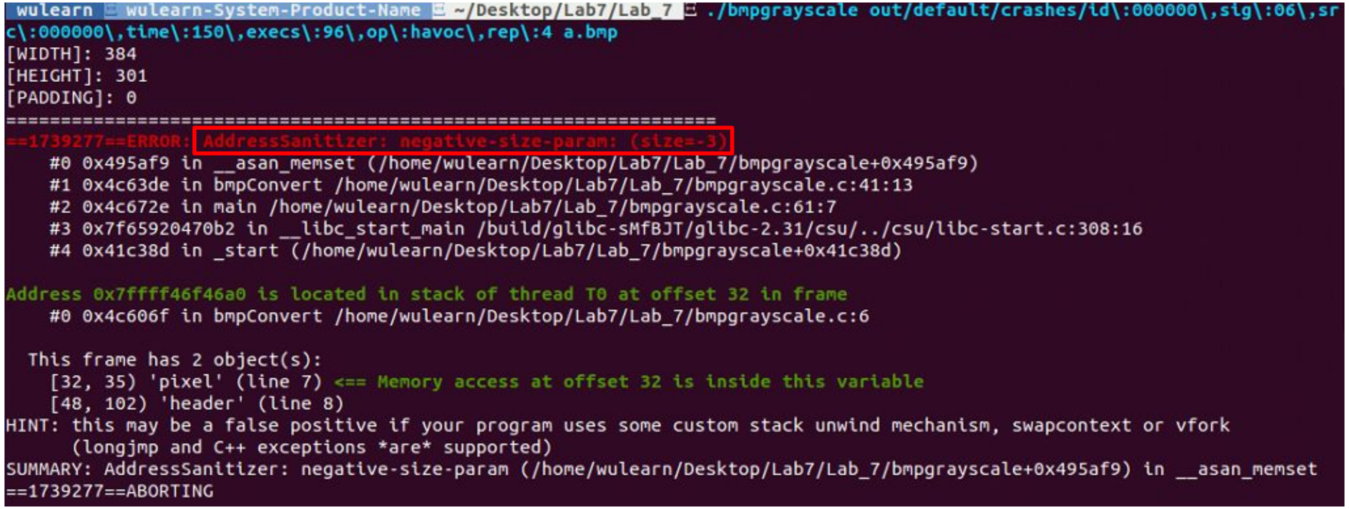


wulearn@wulearn-System-Product-Name: ~/Desktop/Lab7/L...

```
          american fuzzy lop ++4.01a {default} (./bmpgrayscale) [fast]lts
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 1 min, 7 sec       │  cycles done : 1     │
│   last new find : 0 days, 0 hrs, 0 min, 43 sec      │ corpus count : 27    │
│last saved crash : 0 days, 0 hrs, 0 min, 58 sec      │saved crashes : 2     │
│ last saved hang : none seen yet                     │  saved hangs : 0     │
├─ cycle progress ─────────────────────┬─ map coverage─┴──────────────────────┤
│  now processing : 23.1 (85.2%)       │    map density : 41.18% / 76.47%    │
│  runs timed out : 0 (0.00%)          │ count coverage : 30.85 bits/tuple   │
├─ stage progress ─────────────────────┼─ findings in depth ─────────────────┤
│  now trying : havoc                  │ favored items : 4 (14.81%)          │
│ stage execs : 738/883 (83.58%)       │  new edges on : 5 (18.52%)          │
│ total execs : 116k                   │ total crashes : 1191 (2 saved)      │
│  exec speed : 2090/sec               │  total tmouts : 1 (1 saved)         │
├─ fuzzing strategy yields ────────────┴─────────────┬─ item geometry ───────┤
│   bit flips : disabled (default, enable with -D)   │    levels : 2         │
│  byte flips : disabled (default, enable with -D)   │   pending : 20        │
│ arithmetics : disabled (default, enable with -D)   │  pend fav : 0         │
│  known ints : disabled (default, enable with -D)   │ own finds : 26        │
│  dictionary : n/a                                  │  imported : 0         │
│havoc/splice : 28/61.3k, 0/53.9k                    │ stability : 100.00%   │
│py/custom/rq : unused, unused, unused, unused       ├───────────────────────┘
│    trim/eff : 99.93%/75, disabled                  │          [cpu000: 18%]
└────────────────────────────────────────────────────┘
```

Lab 7

```
$ ./bmpgrayscale out/default/crashes/id… a.bmp
```



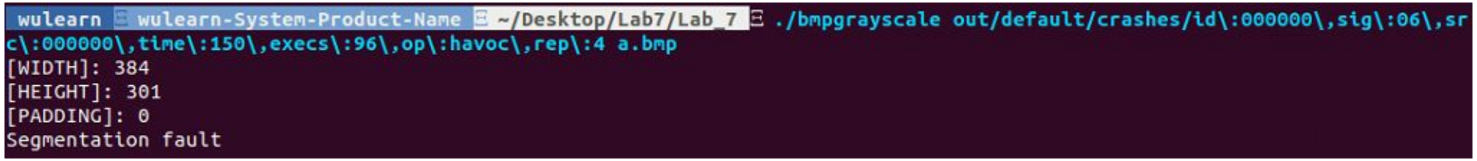
```
wulearn  wulearn-System-Product-Name  ~/Desktop/Lab7/Lab_7  ./bmpgrayscale out/default/crashes/id\:000000\,sig\:06\,sr
c\:000000\,time\:150\,execs\:96\,op\:havoc\,rep\:4 a.bmp
[WIDTH]: 384
[HEIGHT]: 301
[PADDING]: 0
=================================================================
==1739277==ERROR: AddressSanitizer: negative-size-param: (size=-3)
    #0 0x495af9 in __asan_memset (/home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale+0x495af9)
    #1 0x4c63de in bmpConvert /home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale.c:41:13
    #2 0x4c672e in main /home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale.c:61:7
    #3 0x7f65920470b2 in __libc_start_main /build/glibc-sMfBJT/glibc-2.31/csu/../csu/libc-start.c:308:16
    #4 0x41c38d in _start (/home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale+0x41c38d)

Address 0x7ffff46f46a0 is located in stack of thread T0 at offset 32 in frame
    #0 0x4c606f in bmpConvert /home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale.c:6

  This frame has 2 object(s):
    [32, 35) 'pixel' (line 7) <== Memory access at offset 32 is inside this variable
    [48, 102) '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 (/home/wulearn/Desktop/Lab7/Lab_7/bmpgrayscale+0x495af9) in __asan_memset
==1739277==ABORTING
```

Lab 7

```
$ ./bmpgrayscale out/default/crashes/id… a.bmp
```



```
wulearn  wulearn-System-Product-Name  ~/Desktop/Lab7/Lab_7  ./bmpgrayscale out/default/crashes/id\:000000\,sig\:06\,sr
c\:000000\,time\:150\,execs\:96\,op\:havoc\,rep\:4 a.bmp
[WIDTH]: 384
[HEIGHT]: 301
[PADDING]: 0
Segmentation fault
```

Challenge Example

Challenge Example

- If you find a bug with Fuzzing:
    - Report issue

Challenge Example

- If you find a bug with Fuzzing:
    - Report issue
        - If the author confirms:
            - Waiting for the bug to be fixed and patched.
                - Requests CVE ID.
        - If NO (the author has disappeared):
            - Requests CVE ID.

Challenge Example

- If you find a bug with Fuzzing:
  - Report issue
    - If the author confirms:
      - Waiting for the bug to be fixed and patched.
        - Requests CVE ID.
      - Example:
        - Report issue: https://github.com/libsixel/libsixel/issues/25
          - CVE-2021-40656
    - If NO (the author has disappeared):
      - Requests CVE ID.
      - Example:
        - Report issue: https://github.com/saitoha/libsixel/issues/157
          - CVE-2022-27046

Reference

Reference

- https://github.com/google/AFL
- https://github.com/AFLplusplus/AFLplusplus
- https://aflplus.plus/docs/technical_details/
- https://aflplus.plus/docs/tutorials/libxml2_tutorial/