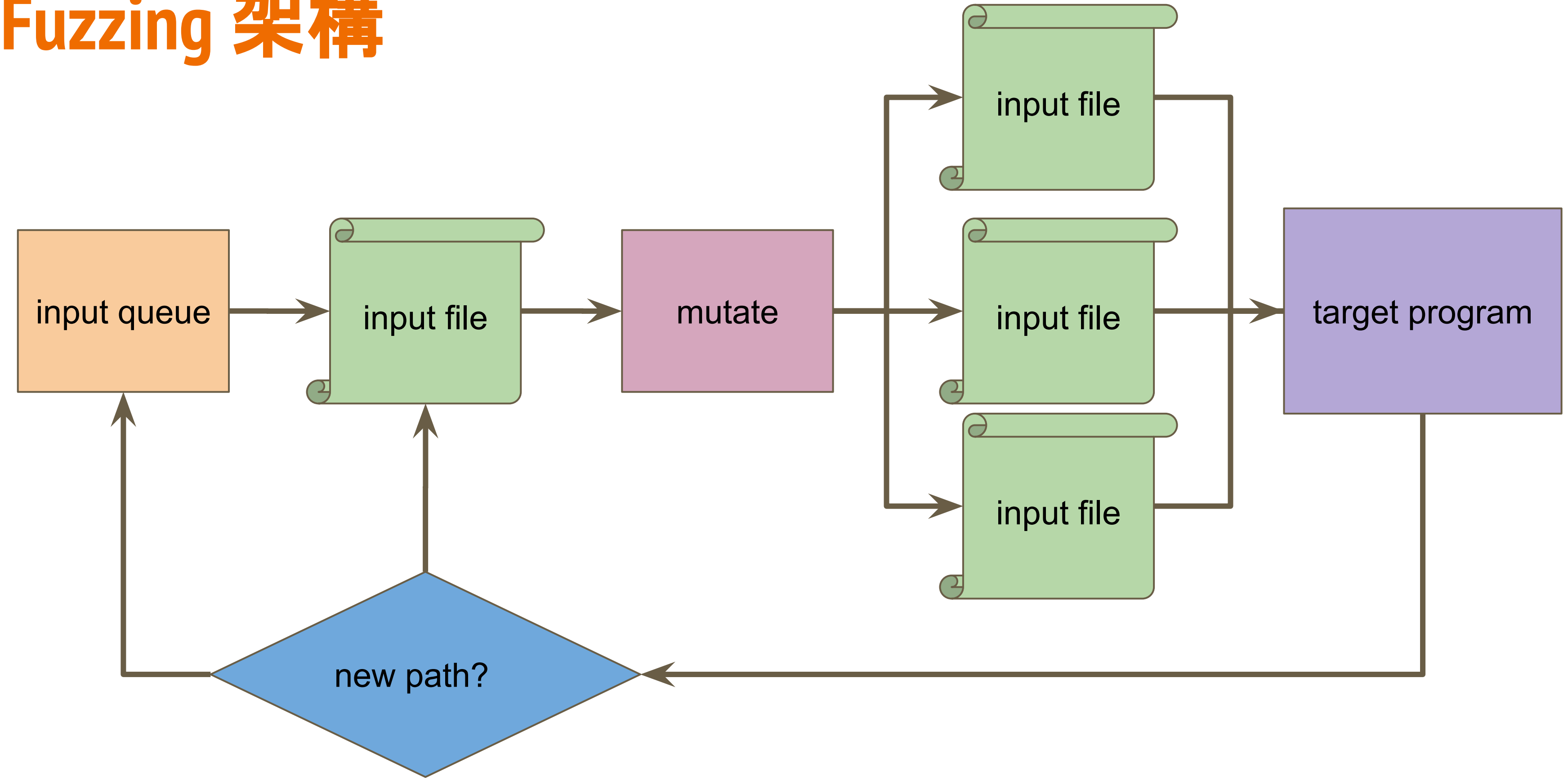
Fuzz Testing

yuan

傳統檢測程式正確

- unit test
- 然而 unit test 往往是人工所寫，很難全部考慮。
    - function 組合是否會發生問題?
    - 不在規格內的輸入是否有作區隔?
    - 一些和內部記憶體配置有關的輸入有無做良好的處理?
- 能不能自動對整支程式做測試？

Fuzzing 架構

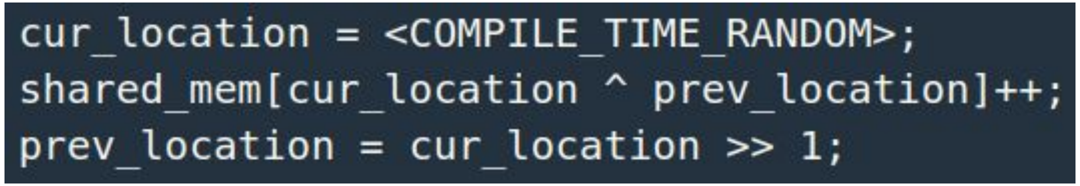

input queue
input file
mutate
input file
input file
input file
target program
new path?

code coverage

- 我們很難去知道 mutate 後的輸入是否是好的?
- 目前最常是根據 code coverage
  - 希望踩到平常沒踩過的 basic block
  - 希望踩到越多 basic block

instrumentation

- 要如何快速去取得程式的執行狀況?
- 有 source code
  - 透過 gcc、clang、LLVM 等編譯工具進行程式碼插樁
    - 在每個 basic block 前面加入特定的程式碼



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

- 沒有 source code
  - Binary 直接改寫
  - 模擬器 (Qemu、Unicorn、Qiling)

mutate

- 透過去變異現有文件來產生輸入
- bitflip x/y : 以每 y bit 翻轉相鄰的 x 個bit
- arithmetic x/y : 以每 y bit 翻轉相鄰的 x 個bit進行加減運算
- interest x/y : 以每 y bit 翻轉相鄰的 x 個bit替換成特定值
  - ex : INT_MAX , 0
- havoc : 對文件大破壞
- splice : 把兩個文件拼接

AFL

- google 所推出
- 覆蓋率回饋指引的先行者
- 但 2017 年後....就沒在更新了
- 於是....

AFL++

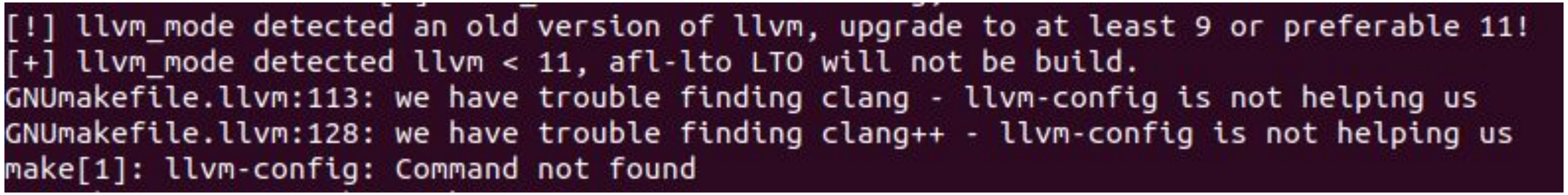
- 廣大的模糊測試社群
- 集合大量優質論文與開源社群的改善
- 持續的更新，並持續結合新的模糊測試技術
  - example : 使用 deep learning，新的變異技術 等等
- https://github.com/AFLplusplus/AFLplusplus

前置作業

- 對於程式碼插樁使用 LLVM 做優化
- LLVM 安裝
  - 先前安裝過了
  - 可以透過 llvm-config-[number] 看電腦有啥版本
- 安裝 gcc 插件
  - sudo apt-get install -y gcc-$(gcc --version|head -n1|sed 's/.* //'|sed 's/\..*//')-plugin-dev libstdc++-$(gcc --version|head -n1|sed 's/.* //'|sed 's/\..*//')-dev
- 如果怕有些沒裝好
  - https://github.com/AFLplusplus/AFLplusplus
  - 看他的 README

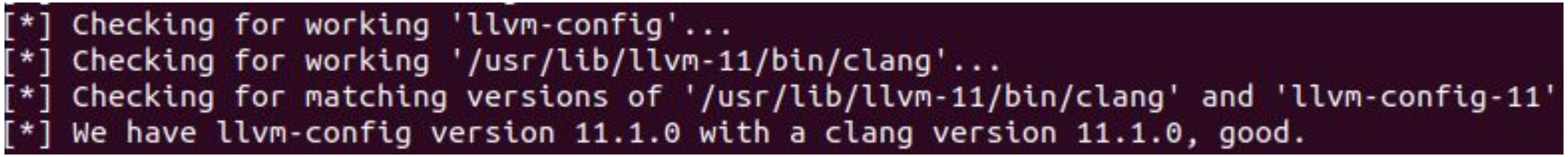
編譯

- $ LLVM_CONFIG=llvm-config-11 make
- LLVM 抓不到或版本太低：



```
[!] llvm_mode detected an old version of llvm, upgrade to at least 9 or preferable 11!
[+] llvm_mode detected llvm < 11, afl-lto LTO will not be build.
GNUmakefile.llvm:113: we have trouble finding clang - llvm-config is not helping us
GNUmakefile.llvm:128: we have trouble finding clang++ - llvm-config is not helping us
make[1]: llvm-config: Command not found
```

成功：



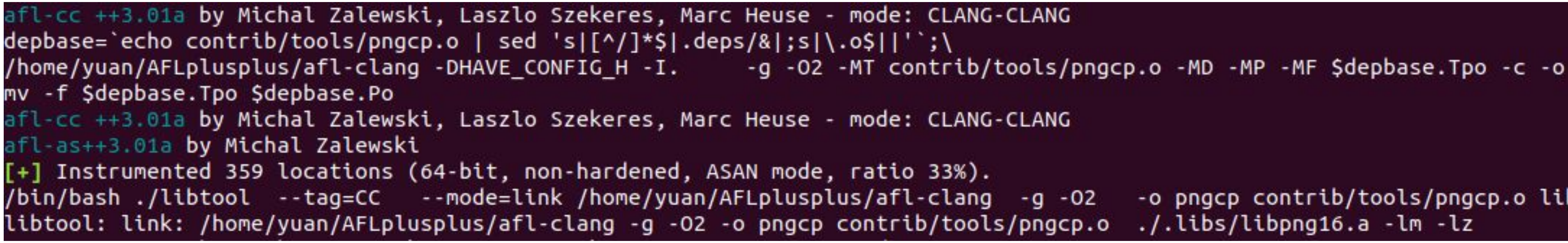
```
[*] Checking for working 'llvm-config'...
[*] Checking for working '/usr/lib/llvm-11/bin/clang'...
[*] Checking for matching versions of '/usr/lib/llvm-11/bin/clang' and 'llvm-config-11'
[*] We have llvm-config version 11.1.0 with a clang version 11.1.0, good.
```

編譯結果

- 插椿的編譯器 :
  - afl-gcc / afl-g++
  - afl-gcc-fast / afl-g++-fast <--- 建議使用，模糊測試速度會比較好
  - afl-clang / afl-clang++
  - afl-clang-fast / afl-clang-fast++ <--- 建議使用，模糊測試速度會比較好
- afl-fuzz : 主要的模糊測試工具
- afl-showmap afl-cmin ..... : 其他的小工具

如何對開源專案進行模糊測試

- 用對 LIBPNG 當範例
  - git clone https://github.com/glennrp/libpng.git
- 對該專案進行插樁
  - $ export CC=~/AFLplusplus/afl-clang
  - $ export CXX=~/AFLplusplus/afl-clang++
  - $ export AFL_USE_ASAN=1
  - $ ./configure --disable-shared
  - $ make



```
afl-cc ++3.01a by Michal Zalewski, Laszlo Szekeres, Marc Heuse - mode: CLANG-CLANG
depbase=`echo contrib/tools/pngcp.o | sed 's|[^/]*$|.deps/&|;s|\.o$||'`;\
/home/yuan/AFLplusplus/afl-clang -DHAVE_CONFIG_H -I.     -g -O2 -MT contrib/tools/pngcp.o -MD -MP -MF $depbase.Tpo -c -o
mv -f $depbase.Tpo $depbase.Po
afl-cc ++3.01a by Michal Zalewski, Laszlo Szekeres, Marc Heuse - mode: CLANG-CLANG
afl-as++3.01a by Michal Zalewski
[+] Instrumented 359 locations (64-bit, non-hardened, ASAN mode, ratio 33%).
/bin/bash ./libtool  --tag=CC   --mode=link /home/yuan/AFLplusplus/afl-clang  -g -O2   -o pngcp contrib/tools/pngcp.o li
libtool: link: /home/yuan/AFLplusplus/afl-clang -g -O2 -o pngcp contrib/tools/pngcp.o  ./.libs/libpng16.a -lm -lz
```

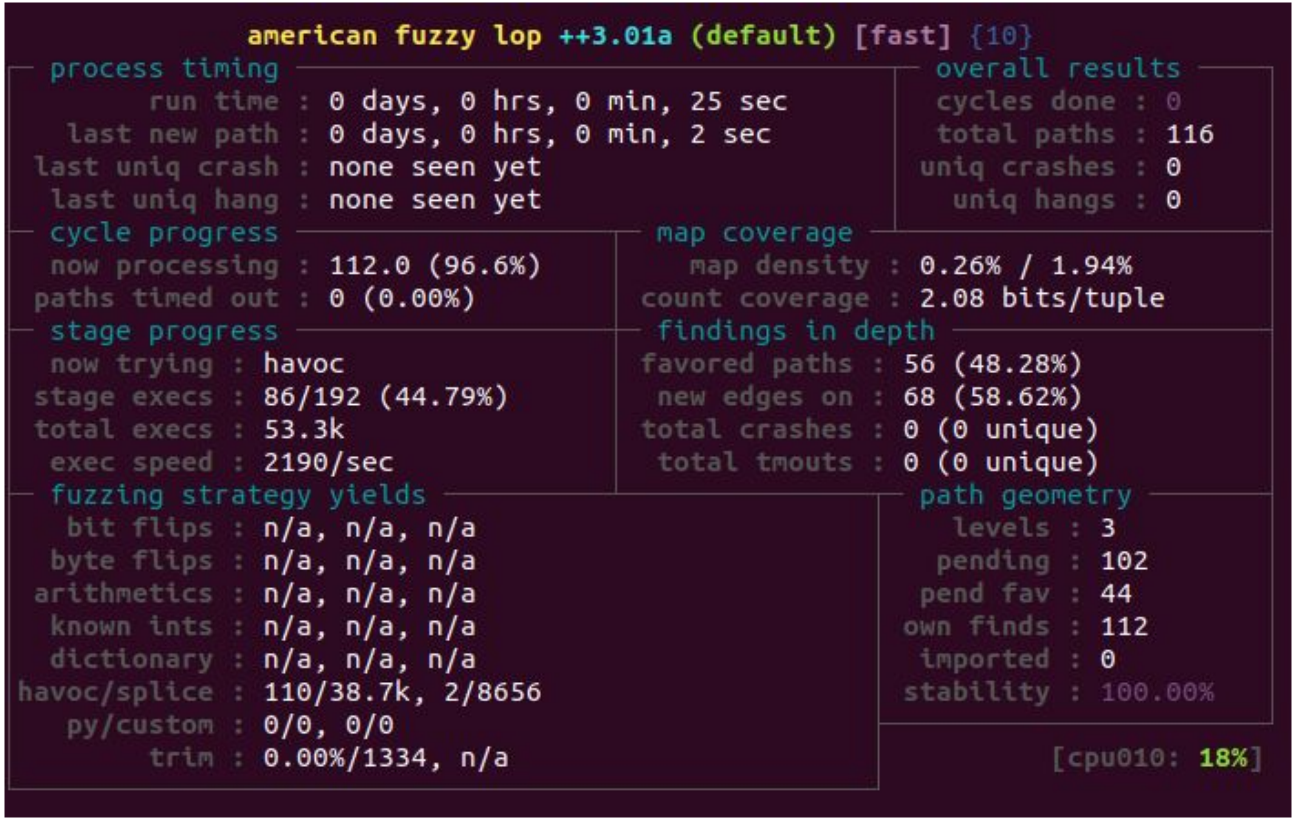
對 cpu 做優化

- sudo ./afl-system-config
- 會關閉 cpu 的節電模式
- 讓 cpu 效能吃到最高（100% core）

模糊測試

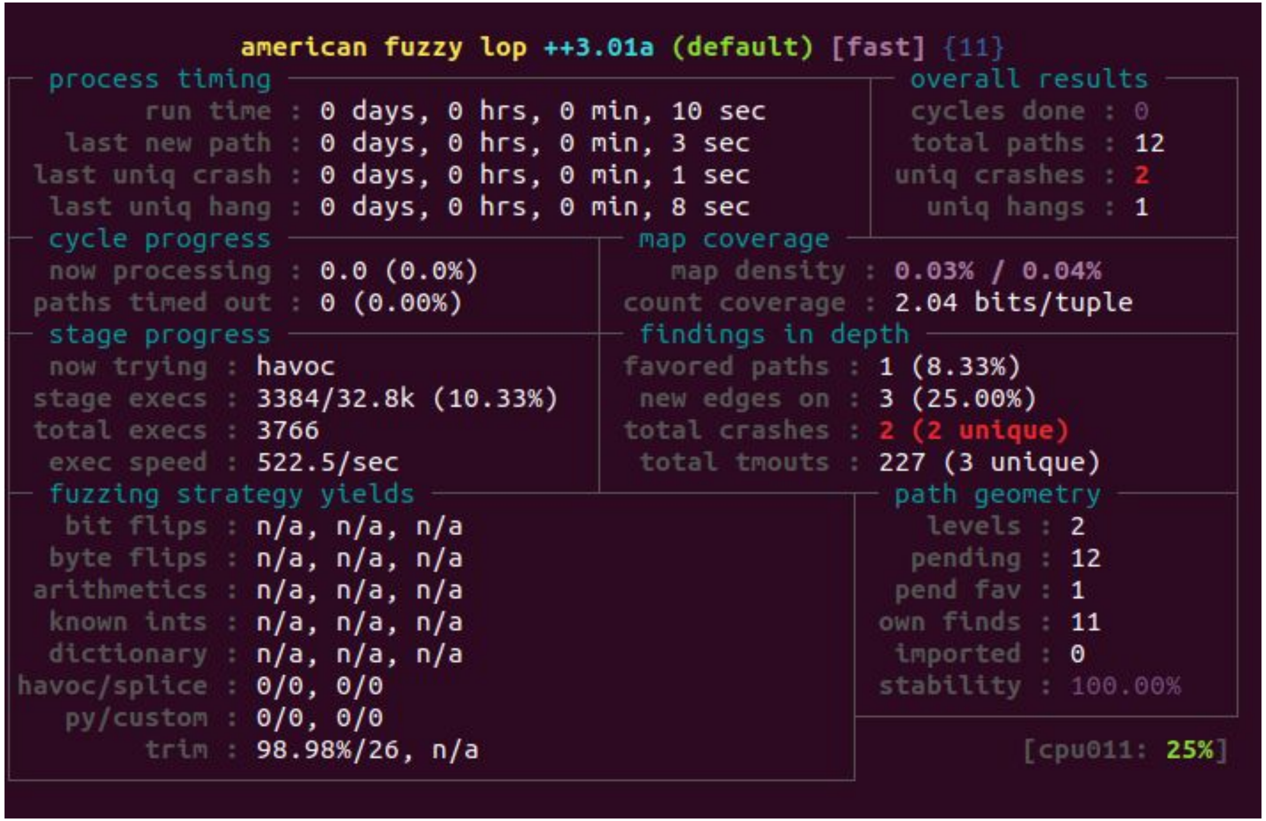
- 針對目標 : libpng/pngimage
  - ./pngimage [png path]
- $ ./afl-fuzz -i ./testcases/images/png -o ./out -b 10 -m none -- ~/libpng/pngimage [argv] @@ [argv]
- -i 最一開始初始 seed 所在資料夾
- -o 模糊測試會產生的資料夾位置
- -b bind 在特定的 cpu core 上
- -m none 不限制使用的記憶體(有 ASAN 才要)
- @@ 要變異的檔案 沒給會直接輸入給 stdin

執行結果



```
              american fuzzy lop ++3.01a (default) [fast] {10}
┌─ process timing ─────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 25 sec        │  cycles done : 0     │
│   last new path : 0 days, 0 hrs, 0 min, 2 sec         │  total paths : 116   │
│ last uniq crash : none seen yet                       │ uniq crashes : 0     │
│  last uniq hang : none seen yet                       │   uniq hangs : 0     │
├─ cycle progress ─────────────────┬─ map coverage ─────┴──────────────────────┤
│  now processing : 112.0 (96.6%)  │    map density : 0.26% / 1.94%           │
│ paths timed out : 0 (0.00%)      │ count coverage : 2.08 bits/tuple         │
├─ stage progress ─────────────────┼─ findings in depth ───────────────────────┤
│  now trying : havoc              │ favored paths : 56 (48.28%)              │
│ stage execs : 86/192 (44.79%)    │  new edges on : 68 (58.62%)              │
│ total execs : 53.3k              │ total crashes : 0 (0 unique)             │
│  exec speed : 2190/sec           │  total tmouts : 0 (0 unique)             │
├─ fuzzing strategy yields ────────┴──────────────┬─ path geometry ───────────┤
│   bit flips : n/a, n/a, n/a                     │    levels : 3             │
│  byte flips : n/a, n/a, n/a                     │   pending : 102           │
│ arithmetics : n/a, n/a, n/a                     │  pend fav : 44            │
│  known ints : n/a, n/a, n/a                     │ own finds : 112           │
│  dictionary : n/a, n/a, n/a                     │  imported : 0             │
│havoc/splice : 110/38.7k, 2/8656                 │ stability : 100.00%       │
│   py/custom : 0/0, 0/0                          ├───────────────────────────┘
│        trim : 0.00%/1334, n/a                   │          [cpu010: 18%]
└─────────────────────────────────────────────────┘
```

執行結果



```
            american fuzzy lop ++3.01a (default) [fast] {11}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 10 sec      │  cycles done : 0     │
│   last new path : 0 days, 0 hrs, 0 min, 3 sec       │  total paths : 12    │
│ last uniq crash : 0 days, 0 hrs, 0 min, 1 sec       │ uniq crashes : 2     │
│  last uniq hang : 0 days, 0 hrs, 0 min, 8 sec       │   uniq hangs : 1     │
├─ cycle progress ──────────────────┬─ map coverage ──┴──────────────────────┤
│  now processing : 0.0 (0.0%)      │    map density : 0.03% / 0.04%         │
│ paths timed out : 0 (0.00%)       │ count coverage : 2.04 bits/tuple       │
├─ stage progress ──────────────────┼─ findings in depth ────────────────────┤
│  now trying : havoc               │ favored paths : 1 (8.33%)              │
│ stage execs : 3384/32.8k (10.33%) │  new edges on : 3 (25.00%)             │
│ total execs : 3766                │ total crashes : 2 (2 unique)           │
│  exec speed : 522.5/sec           │  total tmouts : 227 (3 unique)         │
├─ fuzzing strategy yields ─────────┴───────────────┬─ path geometry ────────┤
│   bit flips : n/a, n/a, n/a                       │    levels : 2          │
│  byte flips : n/a, n/a, n/a                       │   pending : 12         │
│ arithmetics : n/a, n/a, n/a                       │  pend fav : 1          │
│  known ints : n/a, n/a, n/a                       │ own finds : 11         │
│  dictionary : n/a, n/a, n/a                       │  imported : 0          │
│havoc/splice : 0/0, 0/0                            │ stability : 100.00%    │
│  py/custom : 0/0, 0/0                             ├────────────────────────┘
│        trim : 98.98%/26, n/a                      │          [cpu011: 25%]
└───────────────────────────────────────────────────┘
```

輸出結果

- 在所設定的輸出資料夾 /default 內
- crashes <- 程式崩潰的輸入
- hangs <- 程式未在指定時間結束的輸入
  - 可能單純執行時間過長，或是無窮迴圈
- queue <- 變異後有新路徑的輸入

Lab

- 我們提供一個把 bmp 從彩色轉換成灰階的程式
- 裡面有漏洞，幫我用模糊測試找到並修好他。
- https://github.com/iasthc/NYCU-Software-Testing-2021/tree/IOC/Lab-8
- 繳交:學號.zip
- 內容:
  - poc : 會造成問題的輸入
  - bmp_lib.c：你修復好的程式碼
    - 所改的程式碼前面一行加個//Fix
  - 學號.pdf : 裡面寫是啥原因造成這個問題的
- 如果你想公開你修復好的程式碼，麻煩等作業繳交期限截止後，感謝。

Reference

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