Fuzzing – 101

- PHDays 2019

Introduction

Zubin Devnani - www.devtty0.io


p1ngfl0yd

cognosec
A CYBER 1 Company

Dhiraj Mishra - www.inputzero.io


RandomDhiraj

cognosec
A CYBER 1 Company

phd Positive Hack Days

Before we get started

Fuzzing = Patience

- Success mantra

System requirement

Give as much as you can.

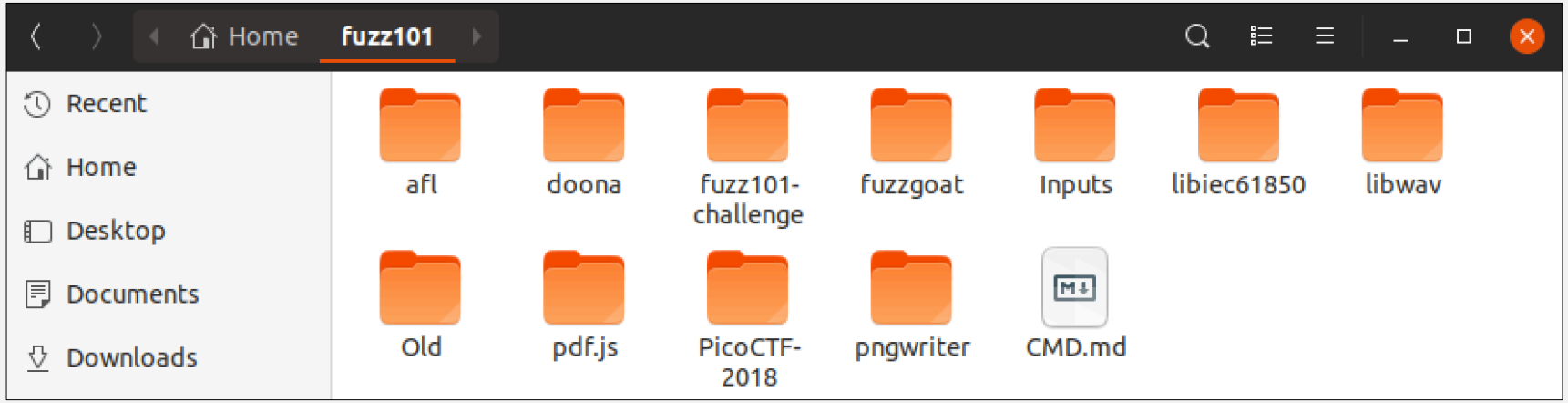
OR

minimum would be

Ubuntu VM, 4GB RAM, 40GB HD

The provided Ubuntu VM

Username: PHDays
Password: E@sy



Agenda

- Not to get bored!!

- Understanding
  - AFL
  - ASAN
  - AFL + ASAN
  - Network Protocol Fuzzing

- Fuzz as much you can for upcoming hours

- Feel free to ask questions

Why fuzzing?

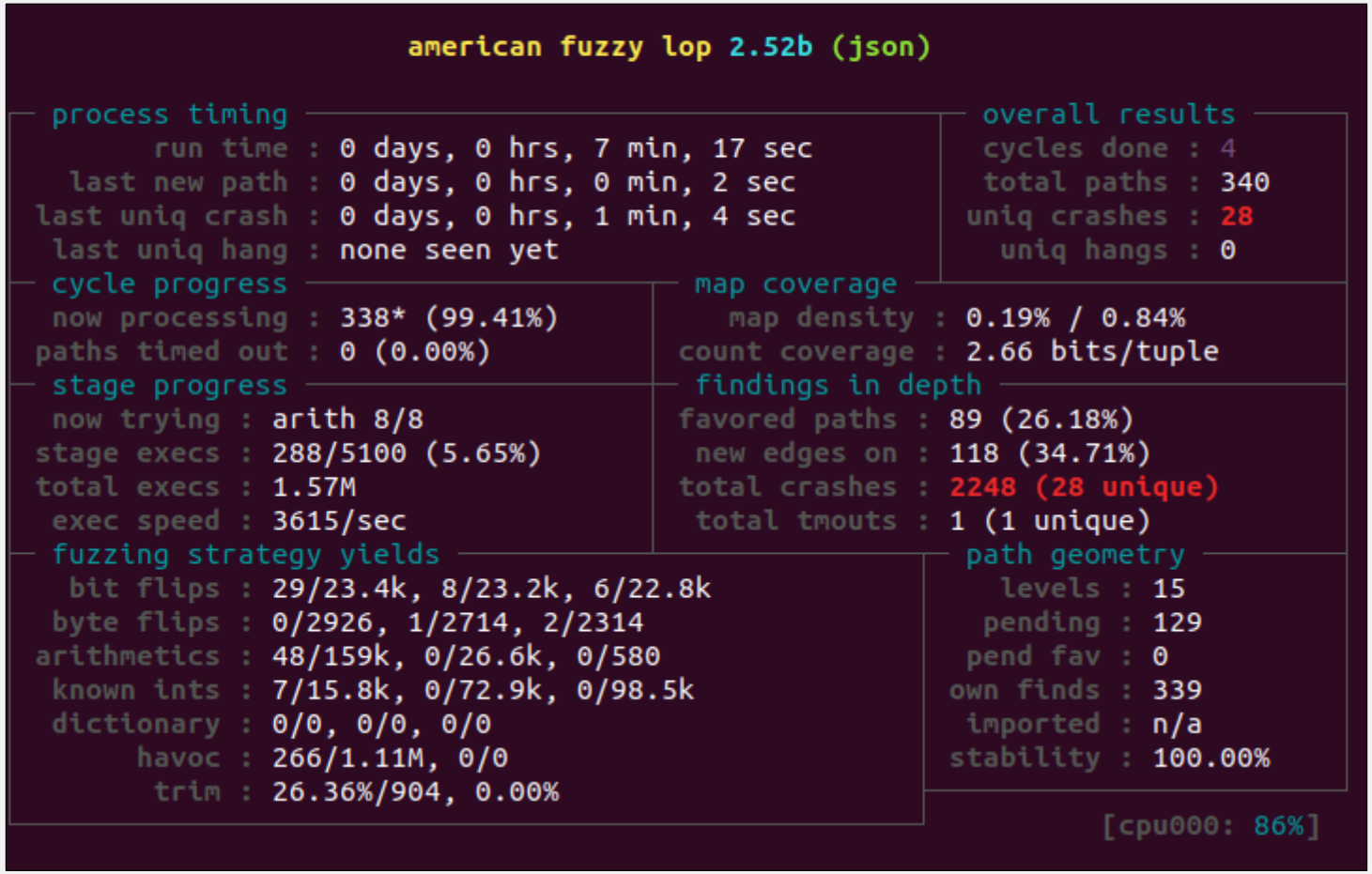
```
==20297==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x629000009748 at pc 0x0000004e58b9 bp 0x7ffca5141520 sp 0x7ffca5140cd0
READ of size 17771 at 0x629000009748 thread T0
    #0 0x4e58b8 in __asan_memcpy /tmp/final/llvm.src/projects/compiler-rt/lib/asan/asan_interceptors_memintrinsics.cc:23:3
    #1 0x5224a8 in tls1_process_heartbeat /home/input0/heartbleed/BUILD/ssl/t1_lib.c:2586:3
    #2 0x58e51d in ssl3_read_bytes /home/input0/heartbleed/BUILD/ssl/s3_pkt.c:1092:4
    #3 0x592c5a in ssl3_get_message /home/input0/heartbleed/BUILD/ssl/s3_both.c:457:7
    #4 0x55e847 in ssl3_get_client_hello /home/input0/heartbleed/BUILD/ssl/s3_srvr.c:941:4
    #5 0x55a8a9 in ssl3_accept /home/input0/heartbleed/BUILD/ssl/s3_srvr.c:357:9
    #6 0x51653d in LLVMFuzzerTestOneInput /home/input0/Downloads/fuzzer-test-suite-master/openssl-1.0.1f/target.cc:34:3
    #7 0x42dd1c in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:515:13
    #8 0x42d57b in fuzzer::Fuzzer::RunOne(unsigned char const*, unsigned long, bool, fuzzer::InputInfo*, bool*) /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:440:3
    #9 0x42efad in fuzzer::Fuzzer::MutateAndTestOne() /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:648:19
    #10 0x42f865 in fuzzer::Fuzzer::Loop(std::vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >,
fuzzer::fuzzer_allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > const&) /tmp/final/llvm.src/projects/compiler-
rt/lib/fuzzer/FuzzerLoop.cpp:775:5
    #11 0x424570 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:754:6
    #12 0x446172 in main /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10
    #13 0x7f9e23b21b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #14 0x41d609 in _start (/home/input0/heartbleed/openssl-1.0.1f-fsanitize_fuzzer+0x41d609)

0x629000009748 is located 0 bytes to the right of 17736-byte region [0x629000005200,0x629000009748)
allocated by thread T0 here:
    #0 0x4e67d3 in __interceptor_malloc /tmp/final/llvm.src/projects/compiler-rt/lib/asan/asan_malloc_linux.cc:88:3
    #1 0x5c1dbb in CRYPTO_malloc /home/input0/heartbleed/BUILD/crypto/mem.c:308:8
    #2 0x594199 in freelist_extract /home/input0/heartbleed/BUILD/ssl/s3_both.c:708:12
    #3 0x594199 in ssl3_setup_read_buffer /home/input0/heartbleed/BUILD/ssl/s3_both.c:770
    #4 0x59477c in ssl3_setup_buffers /home/input0/heartbleed/BUILD/ssl/s3_both.c:827:7
    #5 0x55b474 in ssl3_accept /home/input0/heartbleed/BUILD/ssl/s3_srvr.c:292:9
    #6 0x51653d in LLVMFuzzerTestOneInput /home/input0/Downloads/fuzzer-test-suite-master/openssl-1.0.1f/target.cc:34:3
    #7 0x42dd1c in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:515:13
    #8 0x42f3ad in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >,
fuzzer::fuzzer_allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > const&) /tmp/final/llvm.src/projects/compiler-
rt/lib/fuzzer/FuzzerLoop.cpp:701:3
    #9 0x42f6e5 in fuzzer::Fuzzer::Loop(std::vector<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >,
fuzzer::fuzzer_allocator<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > const&) /tmp/final/llvm.src/projects/compiler-
rt/lib/fuzzer/FuzzerLoop.cpp:739:3
    #10 0x424570 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:754:6
    #11 0x446172 in main /tmp/final/llvm.src/projects/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10
    #12 0x7f9e23b21b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: heap-buffer-overflow /tmp/final/llvm.src/projects/compiler-rt/lib/asan/asan_interceptors_memintrinsics.cc:23:3 in __asan_memcpy
Shadow bytes around the buggy address:
  0x0c527fff9290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c527fff92a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c527fff92b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c527fff92c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c527fff92d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c527fff92e0: 00 00 00 00 00 00 00 00 00[fa]fa fa fa fa fa fa
  0x0c527fff92f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c527fff9300: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c527fff9310: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c527fff9320: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c527fff9330: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==20297==ABORTING
```

Remember Heartbleed?
CVE-2014-0160

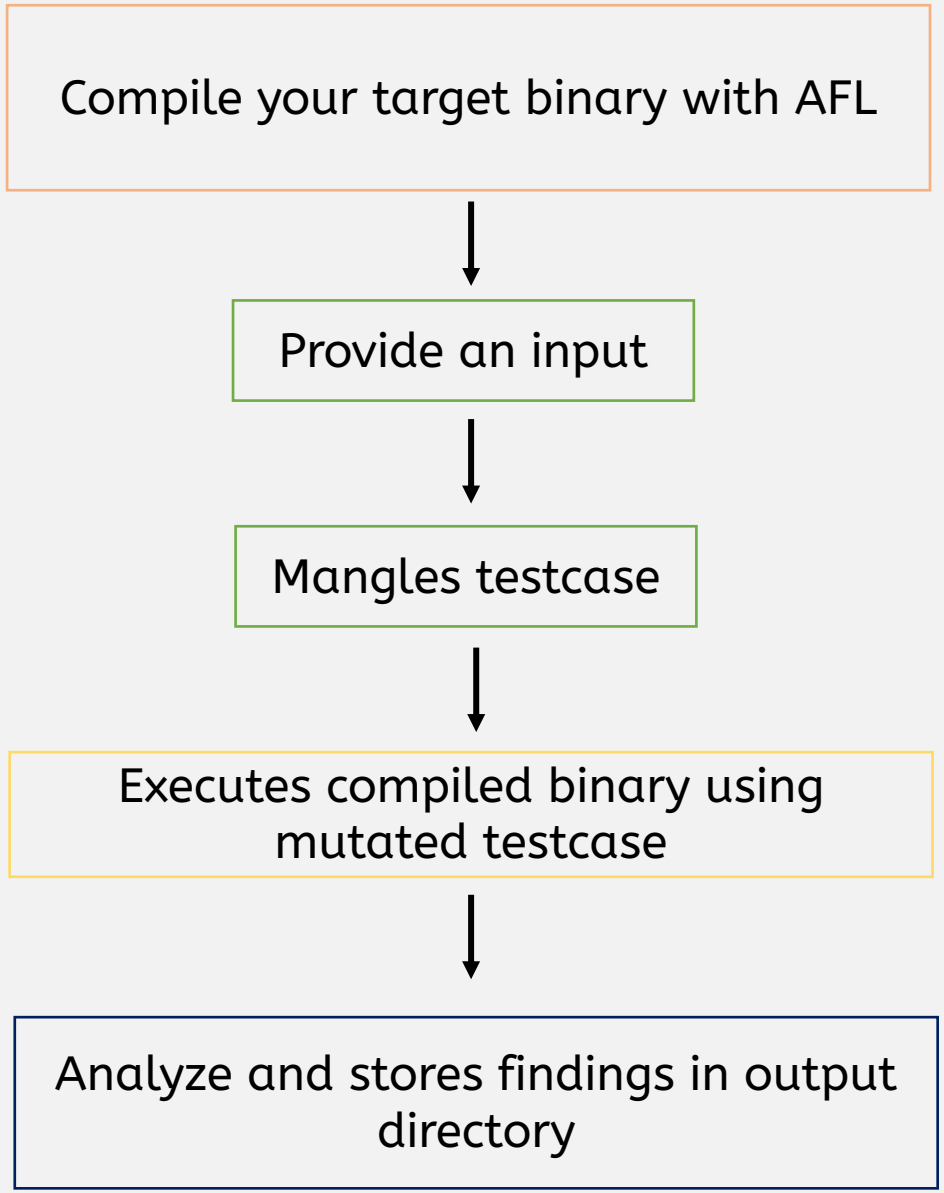
101 - AFL

AFL - American Fuzzy Lop

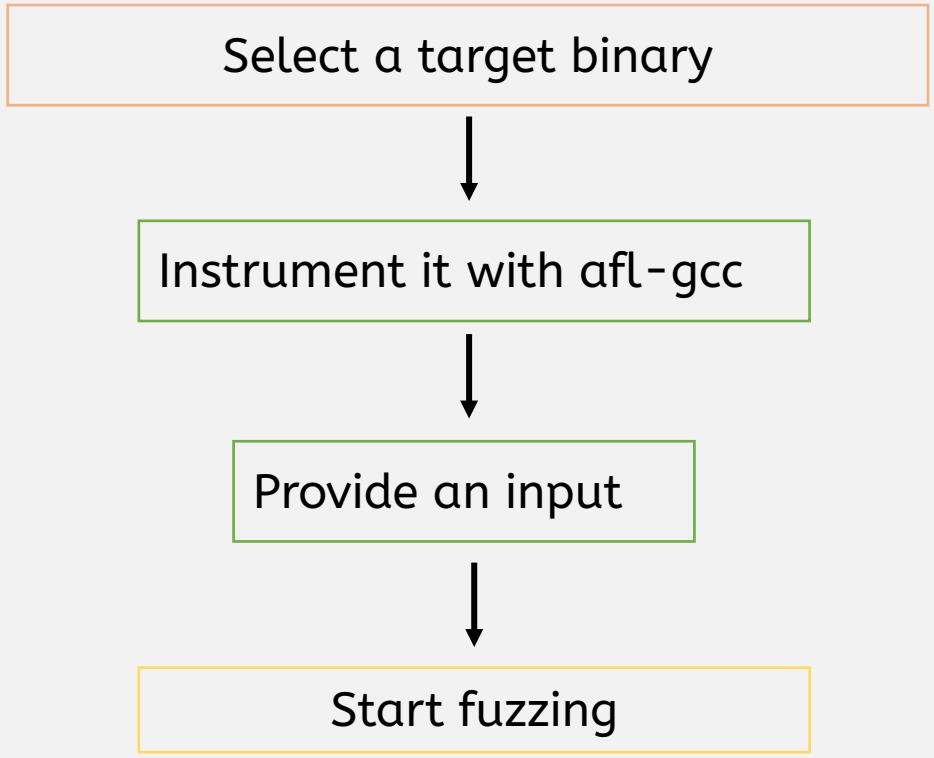
- Developed by - Michal Zalewski
- Mutation based fuzzer
- Input based fuzzer



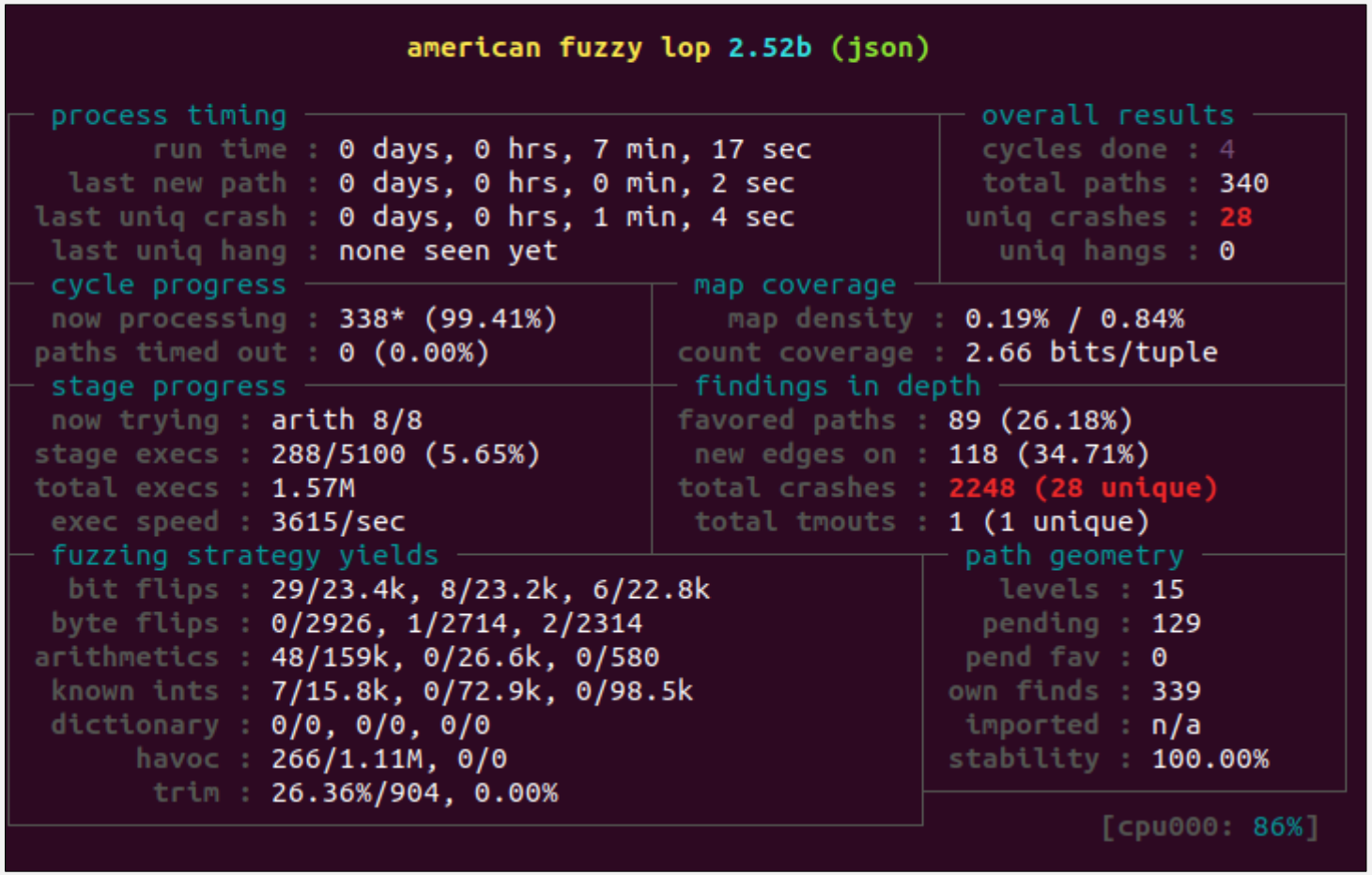
How AFL works?



How to get started?



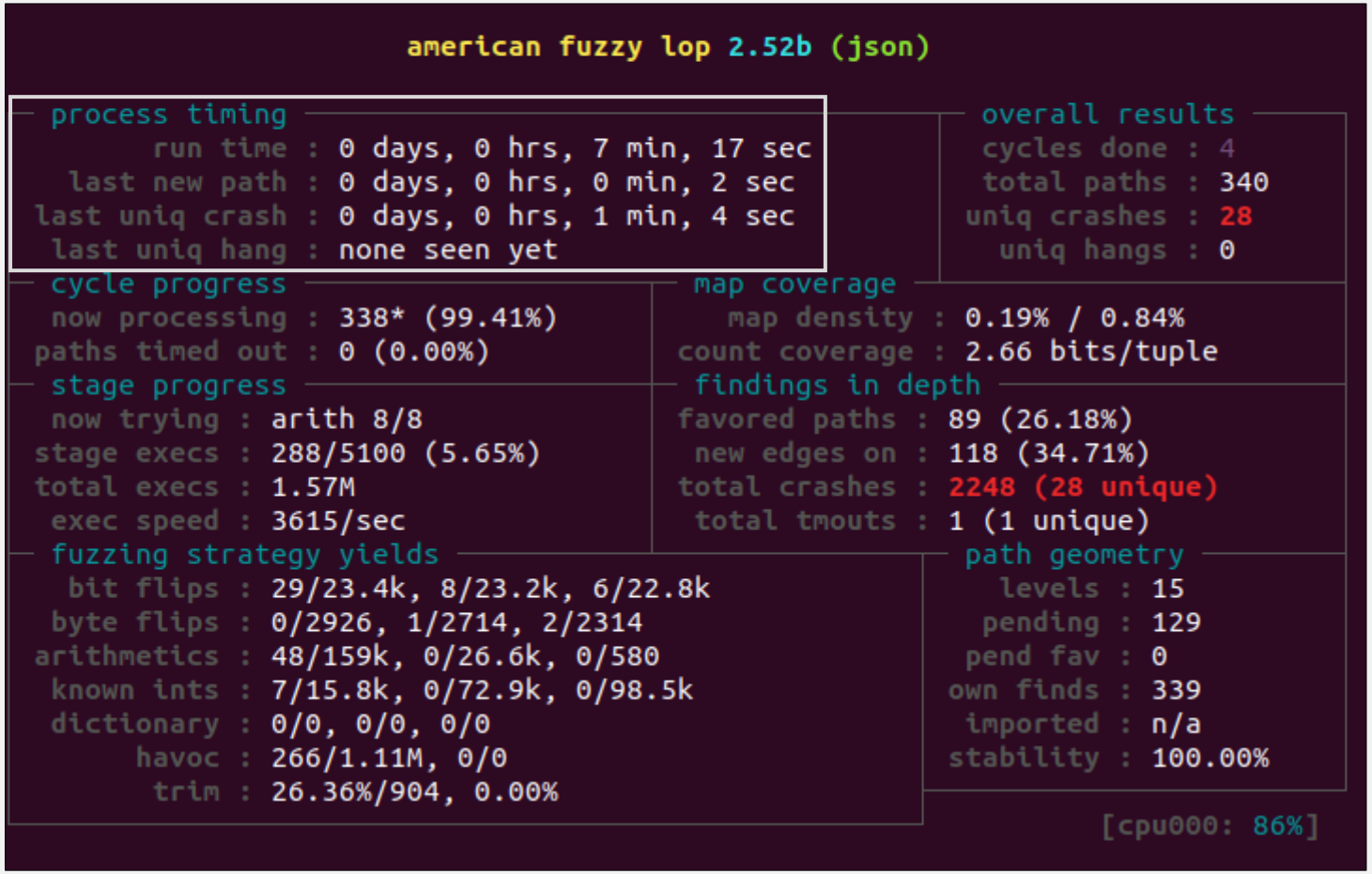
Understanding the color code



Progress and timings:
RED – No progress, very slow, wrong syntax.

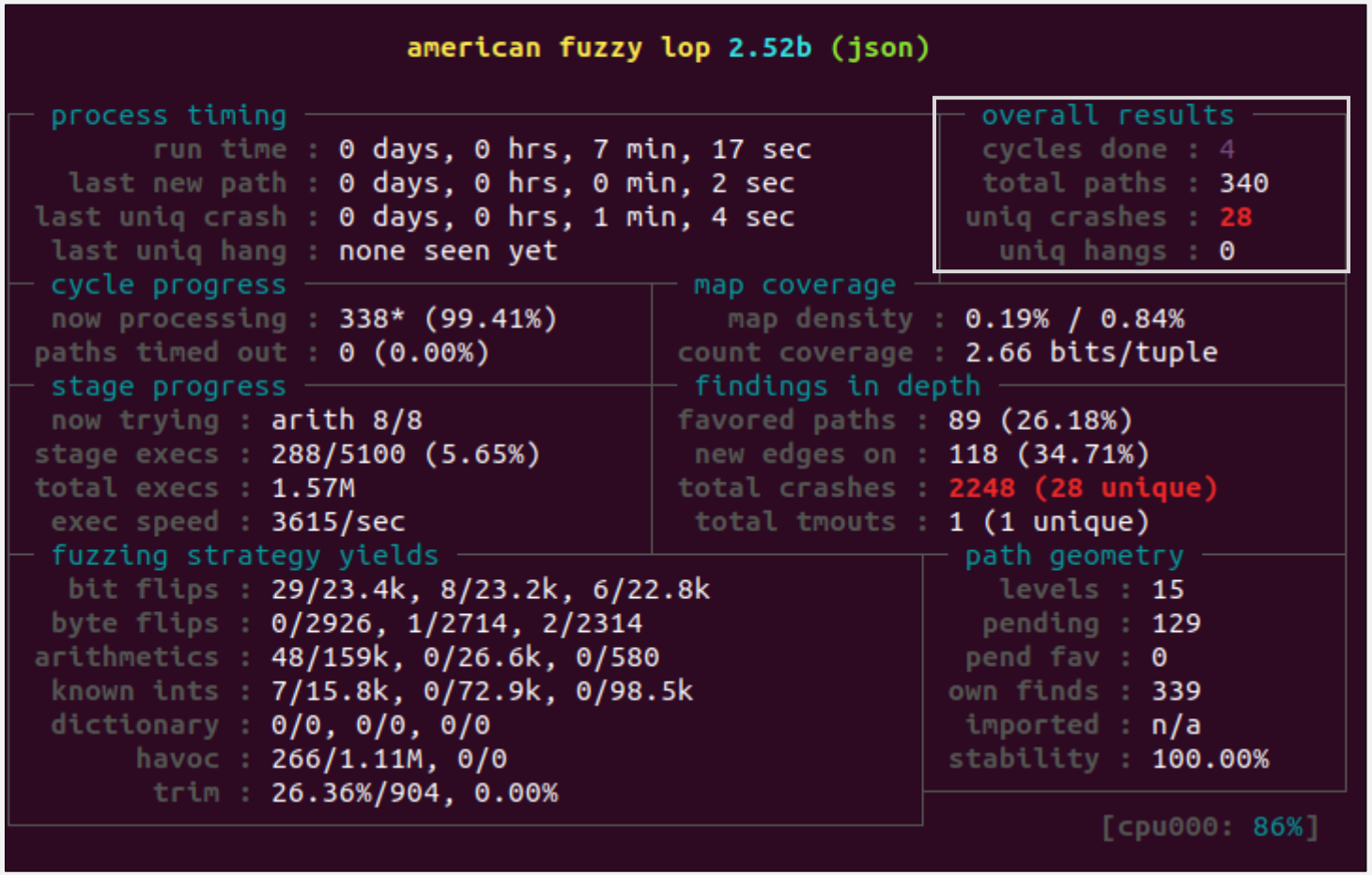
Cycles:
MAGENTA – Started with fuzzing.
YELLOW – New findings.
BLUE – No new finds during last cycle.
GREEN – Large number of cycle was performed.

Understanding the process time



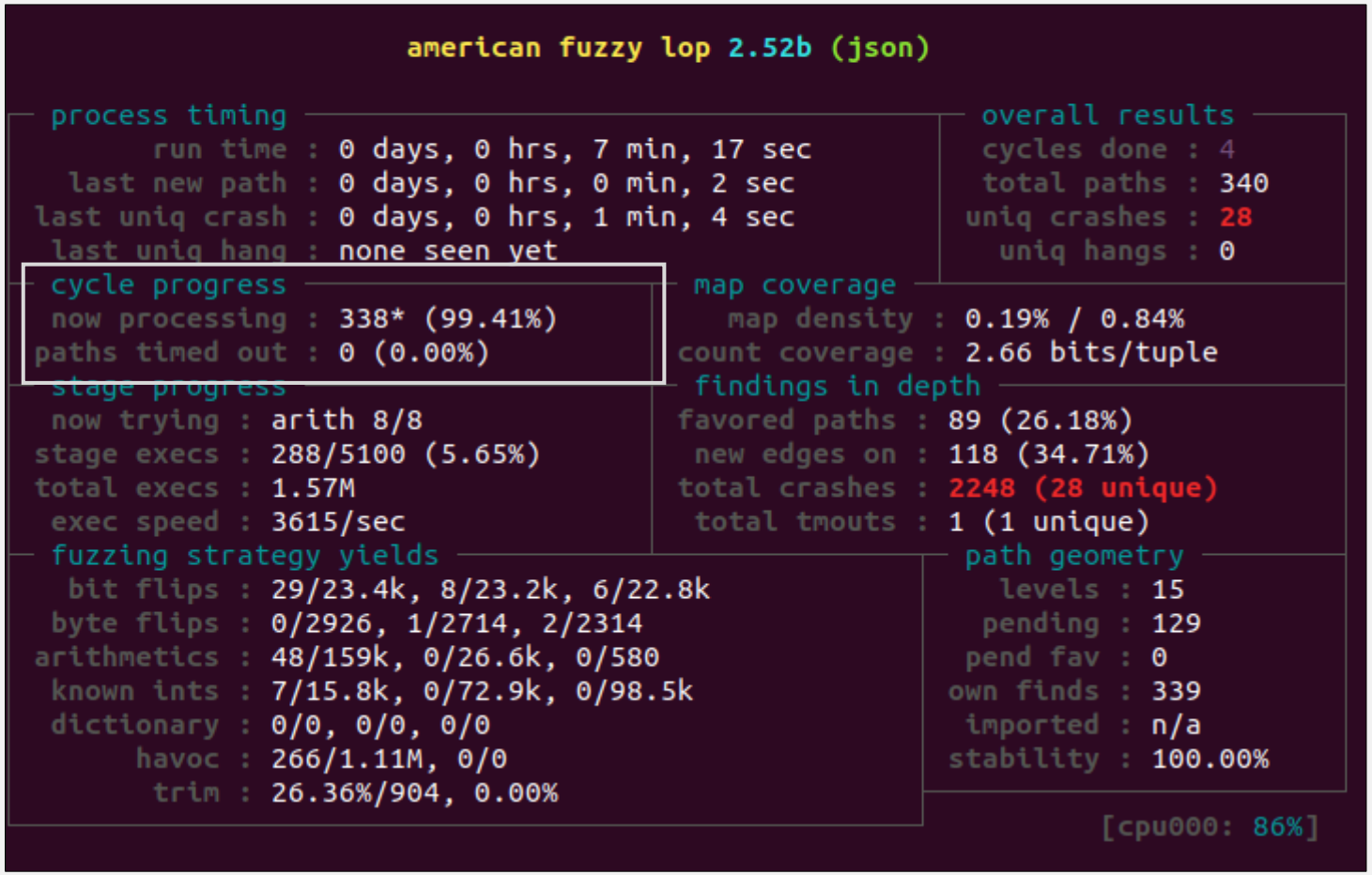
Process timing:
This gives an idea about time elapsed in fuzzing, run time, and last unique crash and hang.

Understanding the overall results



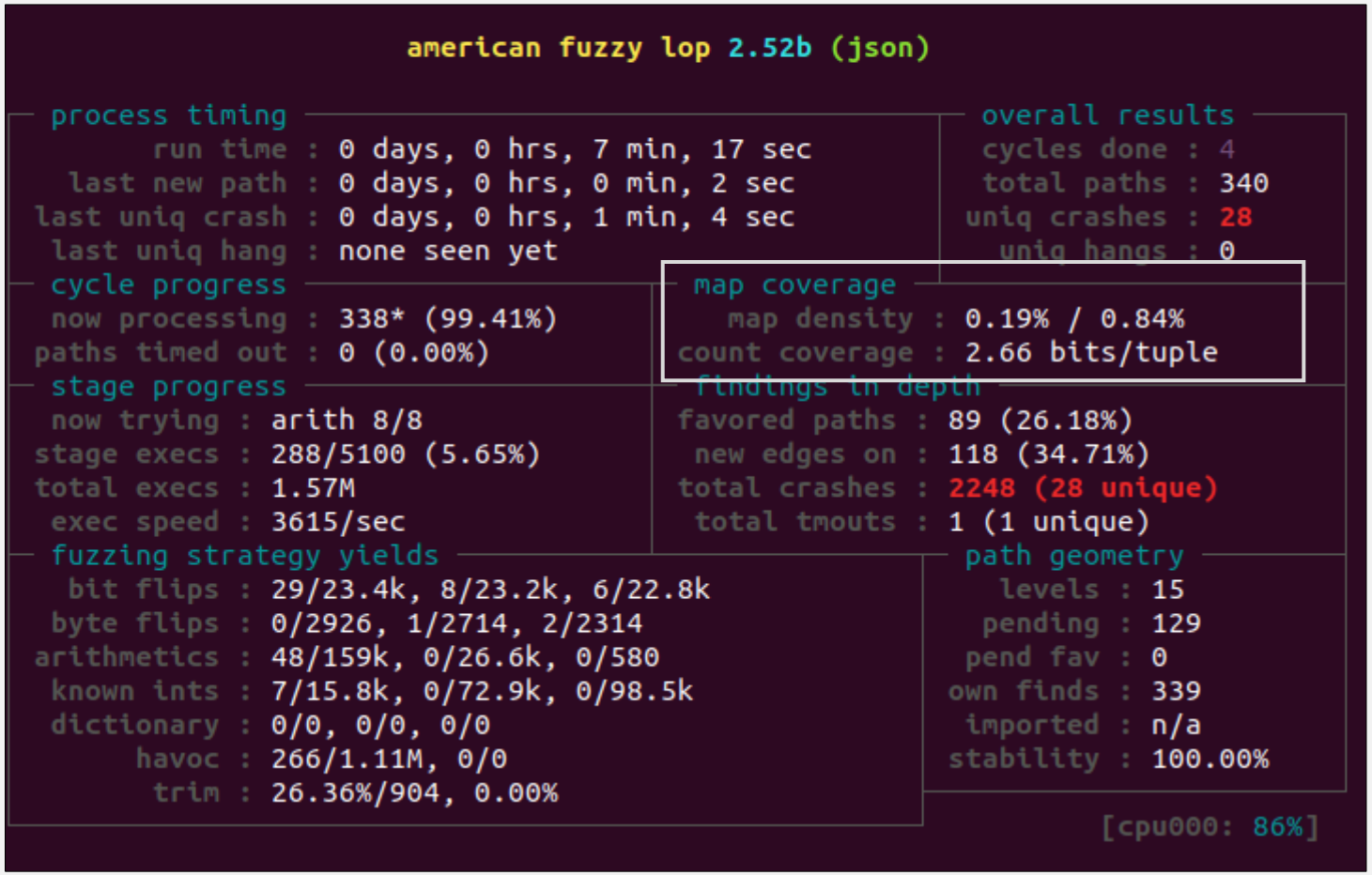
Overall results:
This gives information about cycles done, total path covered so far, and count of unique hangs and crash.

Understanding the cycle progress



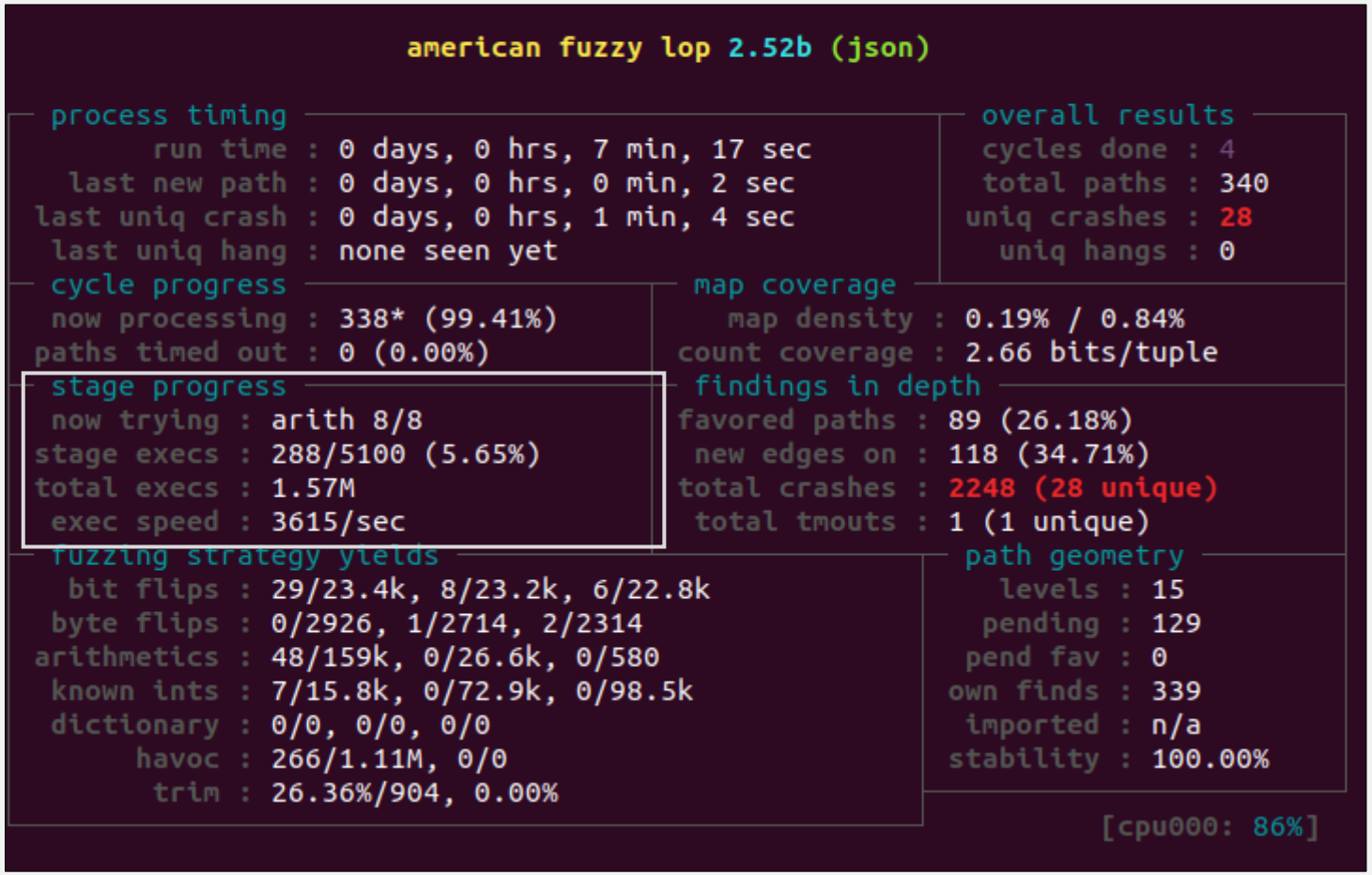
Cycle progress:
ID of current testcase.

Understanding the map coverage



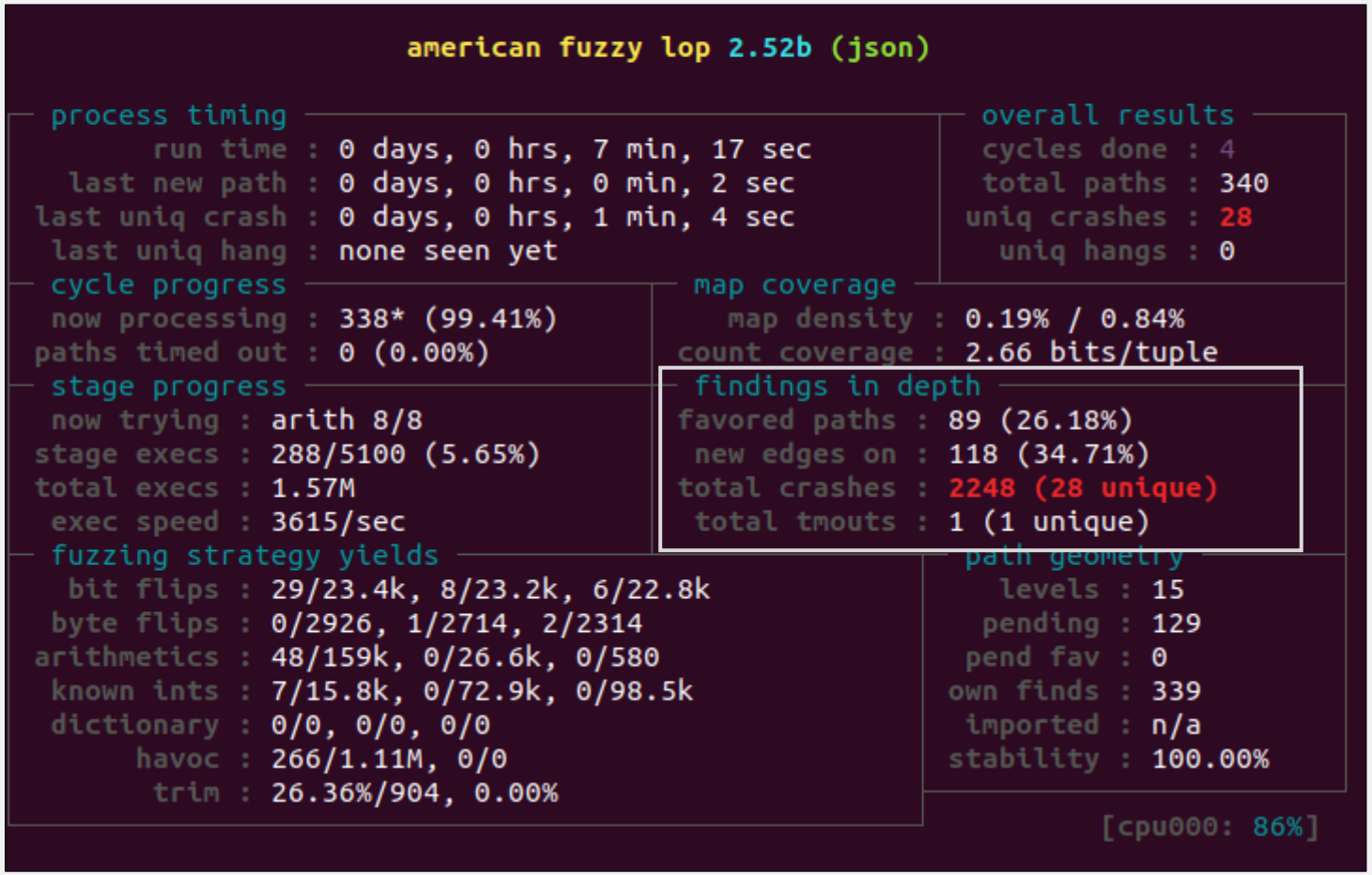
Map coverage:
The section provides some trivia about the coverage observed by the instrumentation embedded in the target binary.

Understanding the stage progress



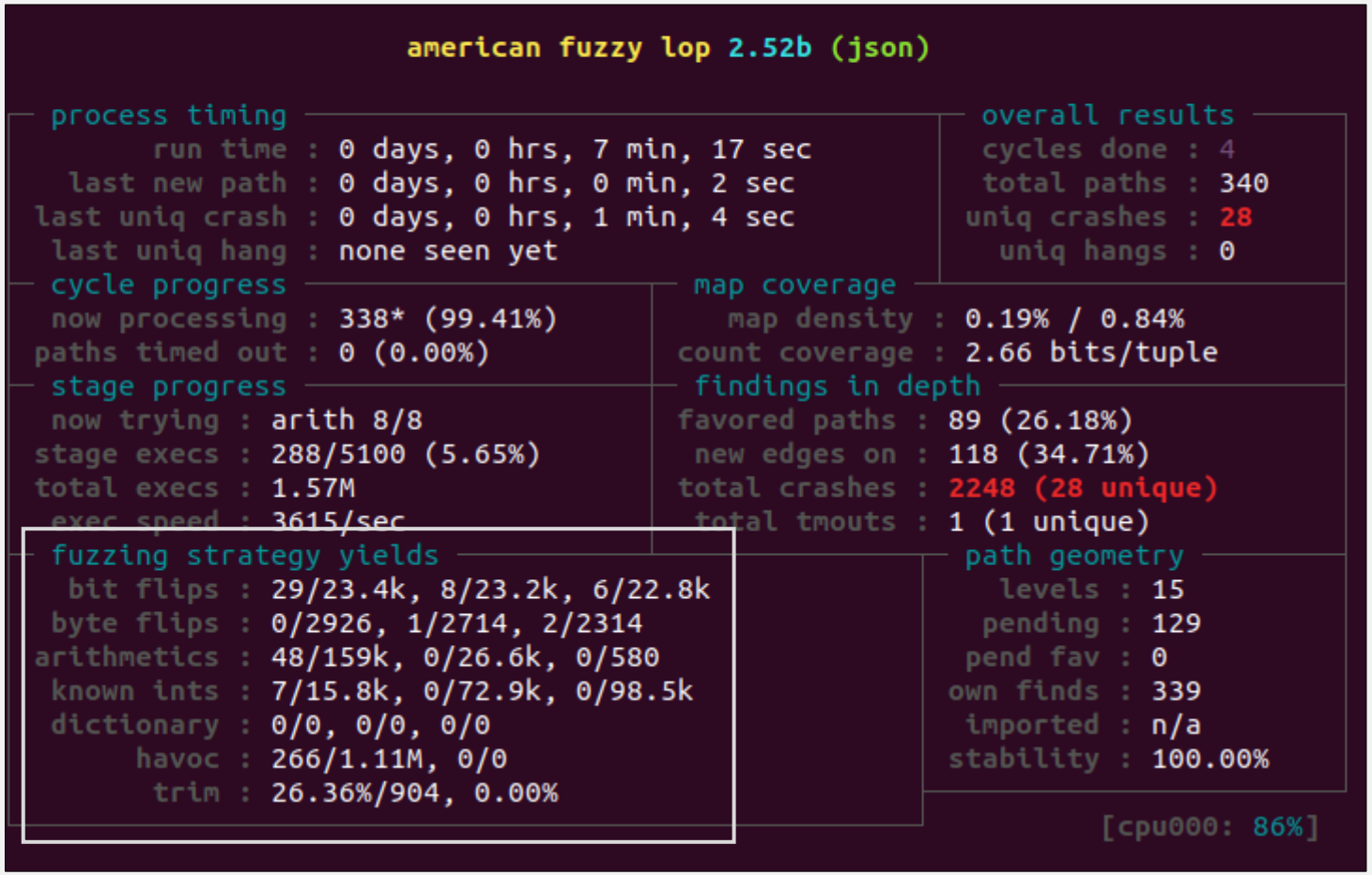
Stage progress:
The section gives an in-depth idea at what the fuzzer is actually doing right now. It has nine core methods which is elaborated further.

Understanding the findings in depth



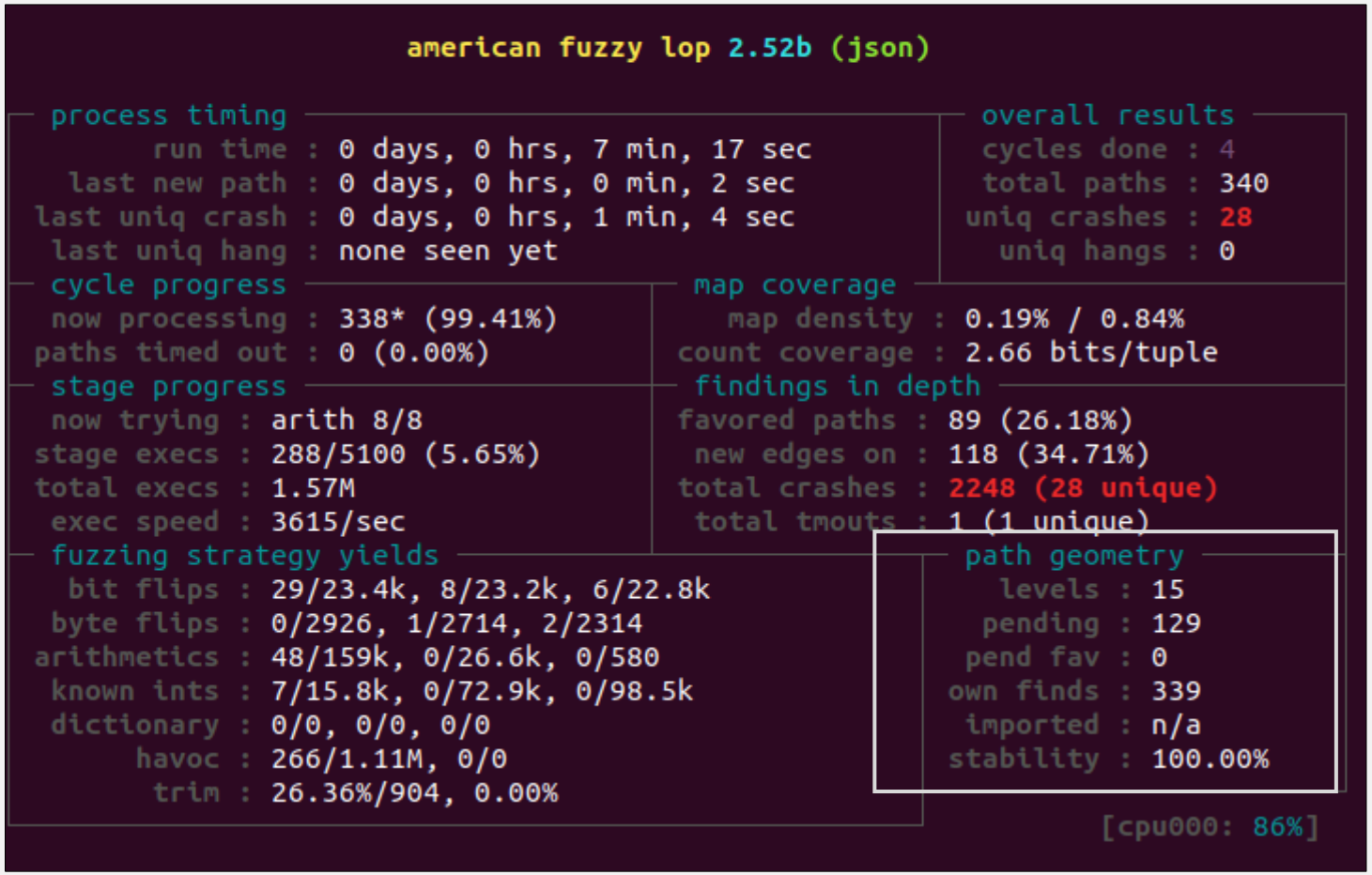
Findings in depth:
Favored paths – select paths on priority ones.
New edges – path results in better edge coverage.
Total crashes and timeouts.

Understanding the fuzzing strategy



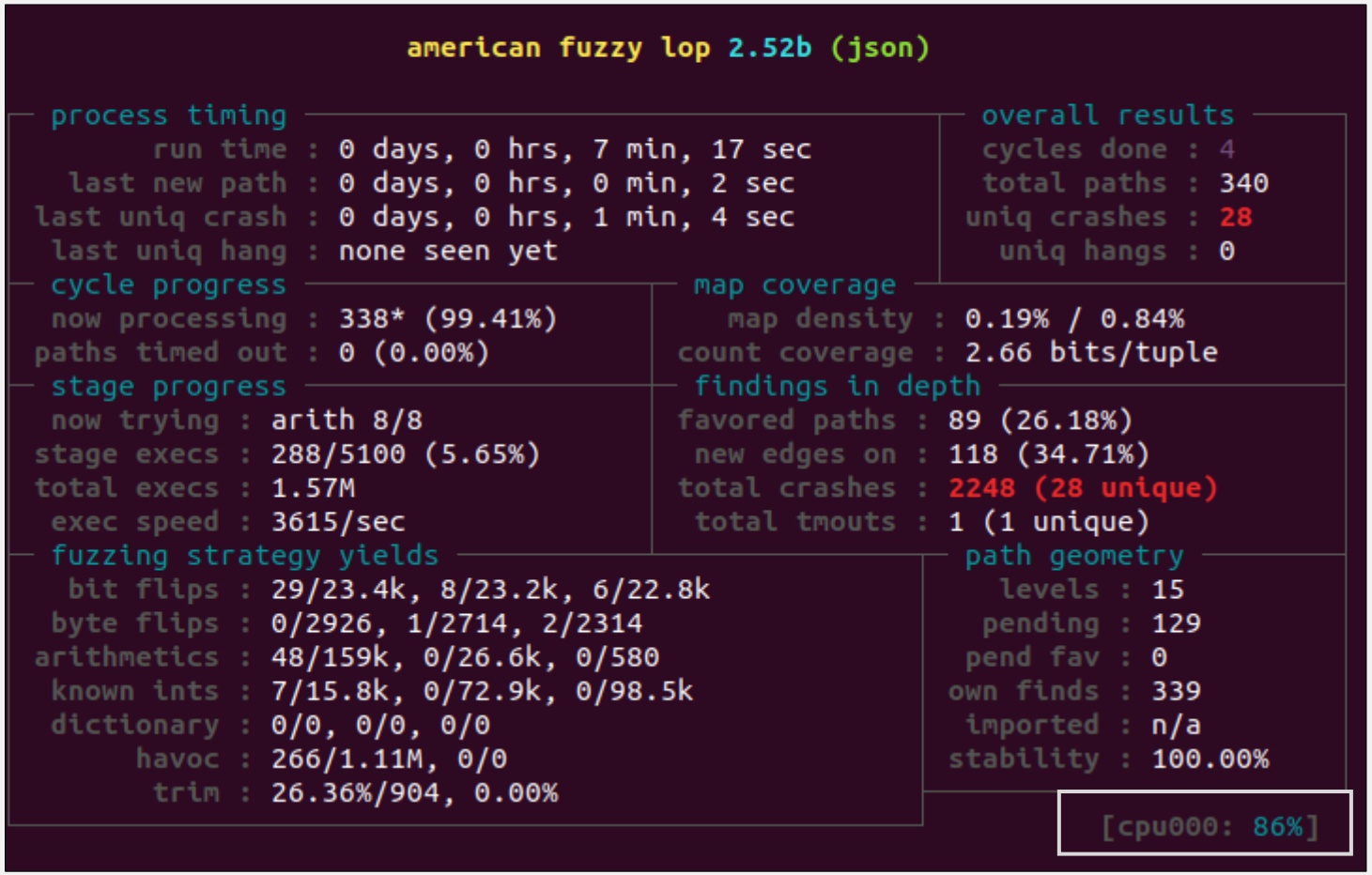
Fuzzing yields:
This is elaborated further in AFL fuzzing strategy

Understanding the path geometry



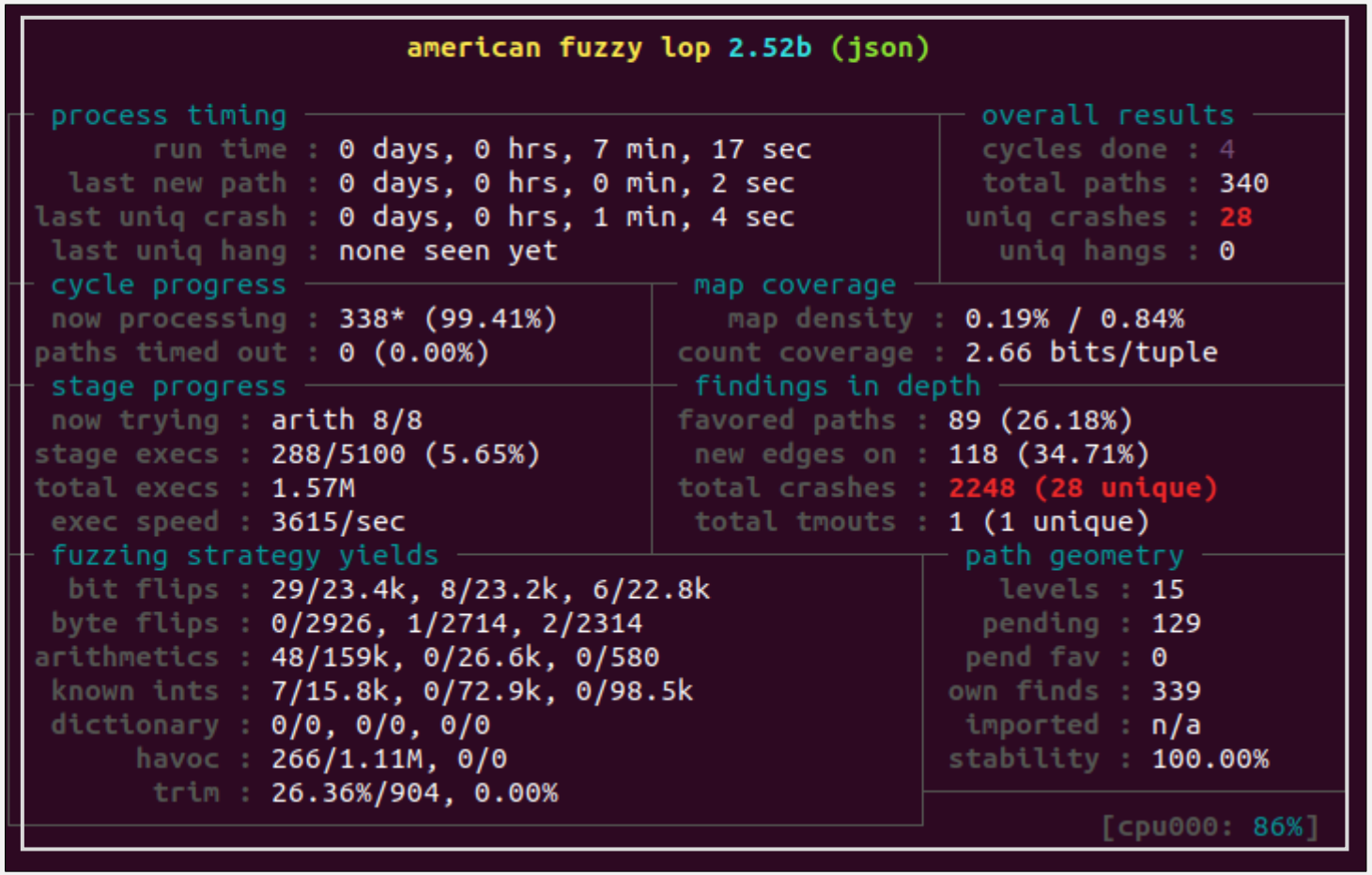
Path geometry:
Levels – Level of initial testcase.
Pending – New testcase which are yet to use in fuzzing.
Pend fav – Pending testcases
Own find – New paths found by fuzzing instance.
Imported – Any paths imported from other fuzzer.
Stability – How stable the fuzzer is while fuzzing the targeted binary.

CPU usage



The term is pretty much self-explanatory, this shows the CPU utilization while fuzzing. (afl-gotcpu)

AFL fuzzing strategy



Calibration - pre-fuzzing stage where the execution path is examined.
Trim - another pre-fuzzing stage where the test case is trimmed to the shortest.
Bitflip - There are number of bits toggled at any given time in input file.
Arith - The fuzzer tries to subtract or add small integers.
Interest - The fuzzer has a list of known interesting bits and values to try.
Extras – User or auto dictionary.
Havoc – Various stack based operations.
Splice - It is equivalent to 'havoc', except that it first splices together two random inputs from the queue.
Sync – Master and slave (Parallel fuzzing).

AFL utilities

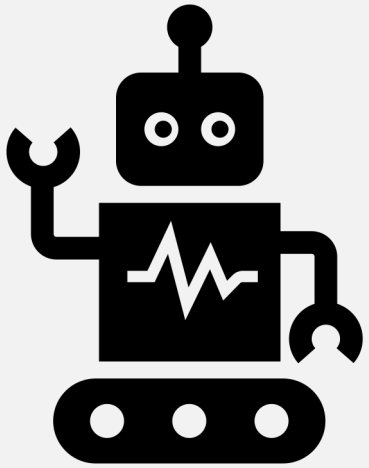
Name	Description
afl-**gcc**	Replacement for gcc, used to recompile binary.
afl-**clang**	Replacement for gcc, used to recompile binary.
afl-**fuzz**	This program takes an binary and attempts a variety of fuzzing strategies.
afl-**cmin**	If a large corpus of data is available for screening, afl-cmin can be used to reject redundant files.
afl-**gotcpu**	Shows CPU utilization.
afl-**showmap**	It runs the targeted binary and displays the contents of the trace bitmap in a human-readable form.
afl-**plot**	It generates gnuplot images from output data.
afl-**tmin**	Test case minimizer
afl-**whatsup**	It checks if the fuzzer is alive.
afl-**analyze**	The tool takes an input file, sequentially flips bytes in this data stream.

Hmmmm…

Name	Description
CC & **CXX**	You will need to override the CC or CXX environment variable before triggering the configure script.
./configure	This script is responsible for getting ready to build the binary and check required dependencies.
--disable-shared	Don’t build shared libraries
Makefile	./configure script produces a customized Makefile specific according to your system, Makefile runs a series of task defined in it.
CMakeList.txt	The file CMakeList.txt is the input to the CMake build system for building software packages.
@@	For programs that take input from a file, use '@@' to mark the location in the target.
echo core > /proc/sys/kernel/core_pattern	It instructs the system to save coredumps as a file instead sending them to system based crash handler app.

Environment pre-requisites

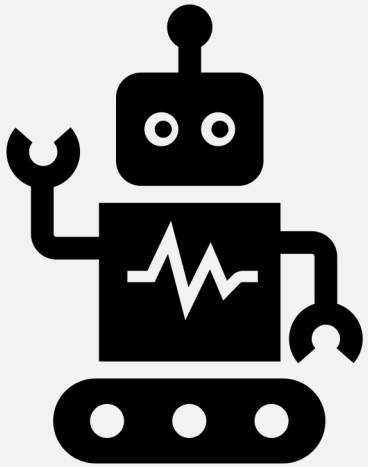
```
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo apt-get install clang
$ sudo apt-get install gcc
$ sudo apt-get install gdb
```



Installing AFL

```
$ sudo apt-get install afl
```

```
$ wget http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz
$ tar xzf afl-latest-X.tgz
$ make
$ sudo make install
```



Demo

Hands down for demo – lets fuzz something!

Fuzzing and analyzing the crashes - libwav

```
input0@zero:~/fuzz101$ git clone https://github.com/marc-q/libwav.git
Cloning into 'libwav'...
remote: Enumerating objects: 100, done.
remote: Total 100 (delta 0), reused 0 (delta 0), pack-reused 100
Receiving objects: 100% (100/100), 27.83 KiB | 150.00 KiB/s, done.
Resolving deltas: 100% (45/45), done.
input0@zero:~/fuzz101$
```

git clone libwav

```
input0@zero:~/fuzz101$ cd libwav/
input0@zero:~/fuzz101/libwav$ ls
libwav.c  libwav.h  LICENSE  README.md  tools
input0@zero:~/fuzz101/libwav$ cd tools/
input0@zero:~/fuzz101/libwav/tools$ ls
wav_gain  wav_info
input0@zero:~/fuzz101/libwav/tools$ cd wav_info/
input0@zero:~/fuzz101/libwav/tools/wav_info$ ls
Makefile  wav_info.c
input0@zero:~/fuzz101/libwav/tools/wav_info$ cat Makefile
CC = gcc
OBJECTS = ../../libwav.c wav_info.c
LIBS =
CFLAGS = -Wall -Wextra -O2
BINDIR = $(DESTDIR)/usr/bin
NAME = wav_info

wav_info: $(OBJECTS)
        $(CC) $(CFLAGS) -o $(NAME) $(OBJECTS) $(LIBS)

clean:
        rm $(NAME)
input0@zero:~/fuzz101/libwav/tools/wav_info$
```

The Makefile uses GNU default compiler `gcc` lets make it to afl-gcc

Fuzzing and analyzing the crashes - libwav

```
input0@zero:~/fuzz101/libwav/tools/wav_info$ cat Makefile
CC = afl-gcc
OBJECTS = ../../libwav.c wav_info.c
LIBS =
CFLAGS = -Wall -Wextra -O2
BINDIR = $(DESTDIR)/usr/bin
NAME = wav_info

wav_info: $(OBJECTS)
        $(CC) $(CFLAGS) -o $(NAME) $(OBJECTS) $(LIBS)

clean:
        rm $(NAME)
input0@zero:~/fuzz101/libwav/tools/wav_info$ make
afl-gcc -Wall -Wextra -O2 -o wav_info ../../libwav.c wav_info.c
afl-cc 2.52b by <lcamtuf@google.com>
afl-as 2.52b by <lcamtuf@google.com>
[+] Instrumented 78 locations (64-bit, non-hardened mode, ratio 100%).
afl-as 2.52b by <lcamtuf@google.com>
[+] Instrumented 8 locations (64-bit, non-hardened mode, ratio 100%).
input0@zero:~/fuzz101/libwav/tools/wav_info$
```

Setting CC to `afl-gcc` and then `make`

```
input0@zero:~/fuzz101/libwav/tools/wav_info$ ./wav_info
LibWAV v. 0.0.1 A (c) 2016 - 2017 Marc Volker Dickmann

Usage: wav_info <filename>!
input0@zero:~/fuzz101/libwav/tools/wav_info$ ls
Makefile  sample.wav  wav_info  wav_info.c
input0@zero:~/fuzz101/libwav/tools/wav_info$ ./wav_info sample.wav
LibWAV v. 0.0.1 A (c) 2016 - 2017 Marc Volker Dickmann

Riff Type:      WAVE
Format:         1
Channels:       2
Samplerate:     44100
Bytespersec:    176400
Blockalign:     4
Bitwidth:       16
input0@zero:~/fuzz101/libwav/tools/wav_info$
```

`wav_info` take .wav as an input to show information about .wav file

Fuzzing and analyzing the crashes - libwav

```
input0@zero:~/fuzz101/libwav/tools/wav_info$ mkdir in ; mkdir out
input0@zero:~/fuzz101/libwav/tools/wav_info$ mv sample.wav in/
```

Setting your corpus

```
input0@zero:~/fuzz101/libwav/tools/wav_info$ afl-fuzz -i in/ -o out/ ./wav_info @@
afl-fuzz 2.52b by <lcamtuf@google.com>
[+] You have 4 CPU cores and 1 runnable tasks (utilization: 25%).
[+] Try parallel jobs - see /usr/share/doc/afl-doc/docs/parallel_fuzzing.txt.
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Checking CPU scaling governor...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning 'in/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:sample.wav'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 351858, map size = 33, exec speed = 669 us
[+] All test cases processed.
```

Started fuzzing with AFL-fuzz

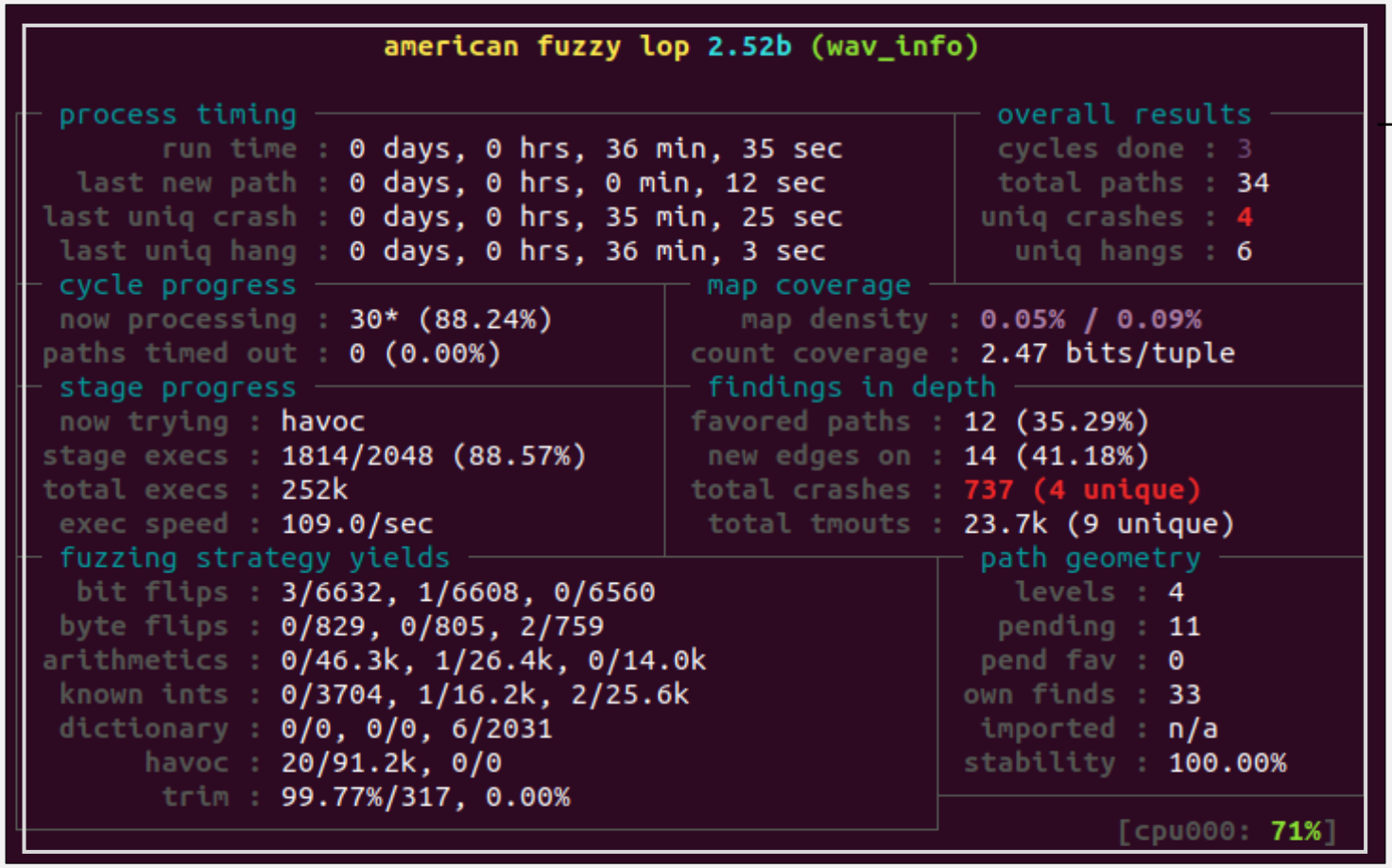
- -i in/ - Input directory
- -o out/ Output directory
- ./wav_info - Binary to fuzz
- @@ - Is used to mark the location in the target's command line where the input file should be in placed.

Understanding the basic checks and tests before fuzzing starts

```
[!] WARNING: Some test cases are huge (343 kB) - see /usr/share/doc/afl-doc/docs/perf_tips.txt!
[+] Here are some useful stats:

    Test case count : 1 favored, 0 variable, 1 total
       Bitmap range : 33 to 33 bits (average: 33.00 bits)
        Exec timing : 669 to 669 us (average: 669 us)

[*] No -t option specified, so I'll use exec timeout of 20 ms.
[+] All set and ready to roll!
```

We can use afl-cmin to minimize the testcases

Fuzzing and analyzing the crashes - libwav



4 unique crashes was found while fuzzing libwav

```
input0@zero:~/fuzz101/libwav/tools/wav_info/out$ ls
crashes  fuzz_bitmap  fuzzer_stats  hangs  plot_data  queue
input0@zero:~/fuzz101/libwav/tools/wav_info/out$
```

Data under out/ (Output) directory

PS: All the four unique crashes are submitted to the vendor by `fouzhe` the credit goes to him.

Fuzzing and analyzing the crashes - libwav

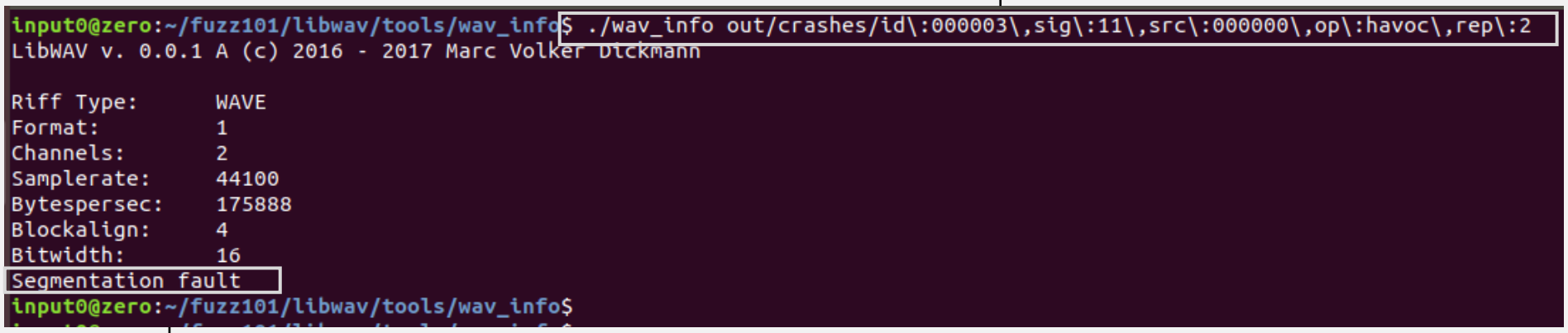
```
input0@zero:~/fuzz101/afl/experimental/crash_triage$ ls
triage_crashes.sh
input0@zero:~/fuzz101/afl/experimental/crash_triage$ ./triage_crashes.sh
crash triage utility for afl-fuzz by <lcamtuf@google.com>

Usage: ./triage_crashes.sh /path/to/afl_output_dir /path/to/tested_binary [...target params...]

input0@zero:~/fuzz101/afl/experimental/crash_triage$
```

Using `.triage_crashes.sh` to analyze the crashes from the output directory

```
input0@zero:~/fuzz101/afl/experimental/crash_triage$ ./triage_crashes.sh /home/input0/fuzz101/libwav/tools/wav_info/out/ /home/input0/fuzz101/
libwav/tools/wav_info/wav_info
crash triage utility for afl-fuzz by <lcamtuf@google.com>


+++ ID 000000, SIGNAL 11 +++


warning: ~/peda/peda.py: No such file or directory
/build/gdb-GT4MLW/gdb-8.1/gdb/utils.c:778: internal-error: virtual memory exhausted: can't allocate 208896 bytes.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Quit this debugging session? (y or n) [answered Y; input not from terminal]

This is a bug, please report it.  For instructions, see:
<http://www.gnu.org/software/gdb/bugs/>.

/build/gdb-GT4MLW/gdb-8.1/gdb/utils.c:778: internal-error: virtual memory exhausted: can't allocate 208896 bytes.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Create a core file of GDB? (y or n) [answered Y; input not from terminal]
Aborted (core dumped)

+++ ID 000001, SIGNAL 11 +++


warning: ~/peda/peda.py: No such file or directory
/build/gdb-GT4MLW/gdb-8.1/gdb/utils.c:778: internal-error: virtual memory exhausted: can't allocate 208896 bytes.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Quit this debugging session? (y or n) [answered Y; input not from terminal]

This is a bug, please report it.  For instructions, see:
<http://www.gnu.org/software/gdb/bugs/>.

/build/gdb-GT4MLW/gdb-8.1/gdb/utils.c:778: internal-error: virtual memory exhausted: can't allocate 208896 bytes.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
```

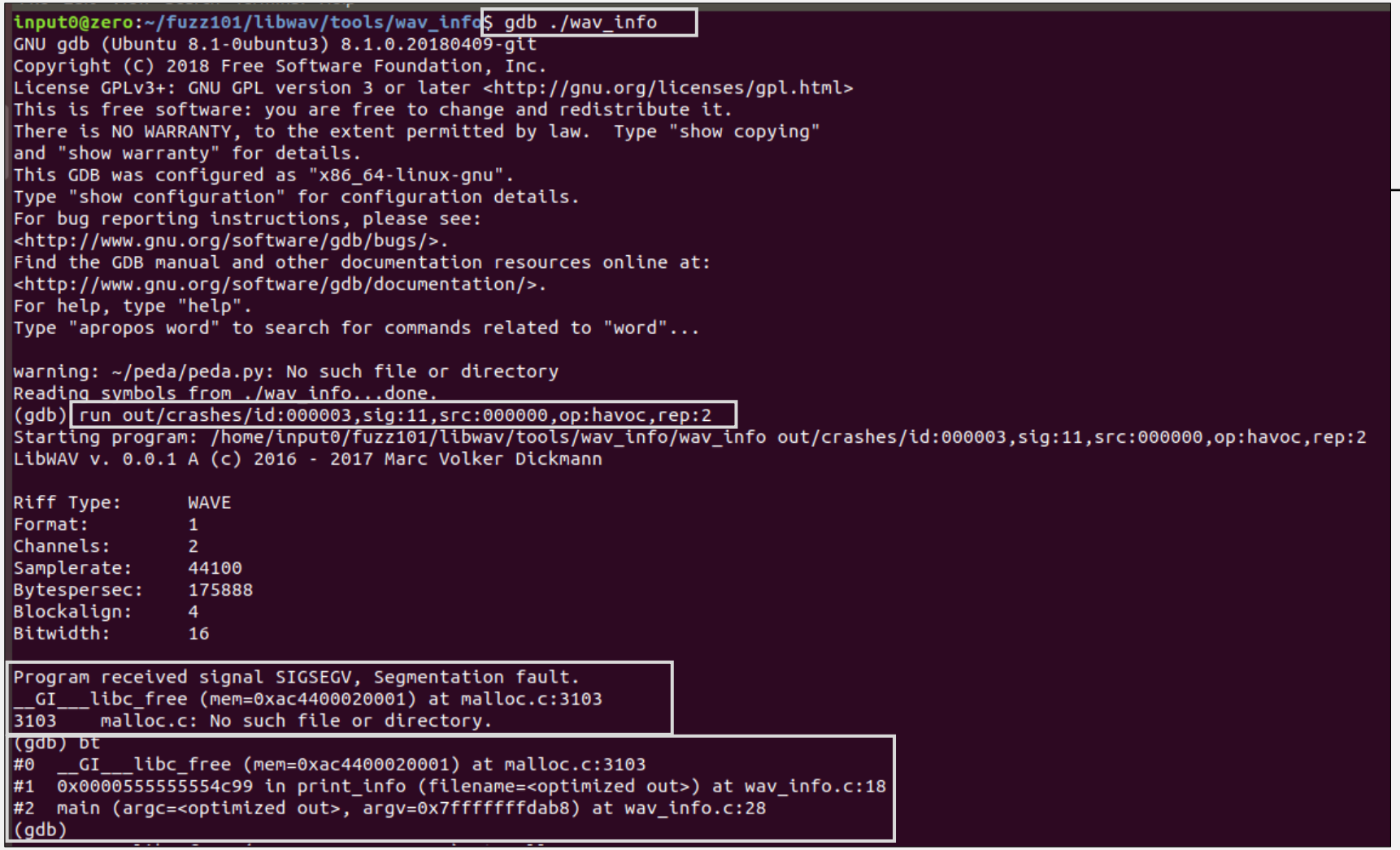
Fuzzing and analyzing the crashes - libwav

Running crashes ID directly



In this case the passed input causes SIGV in `wav_info`

Fuzzing and analyzing the crashes - libwav



Analyzing the crash via GDB

Fuzzing and analyzing the crashes - libwav

```
6  print_info (const char *filename)
7  {
8          wav_file wavfile;
9
10         if (wav_read (&wavfile, filename) != WAV_OK)
11         {
12                 printf ("Error: Couldn't open the file!\n");
13                 return;
14         }
15
16         wav_header_print (&wavfile.header);
17         wav_format_print (&wavfile.format);
18         wav_free (&wavfile);
19 }
20
21 int
22 main (int argc, char *argv[])
23 {
24         printf ("LibWAV v. 0.0.1 A (c) 2016 - 2017 Marc Volker Dickmann\n\n");
25
26         if (argc == 2)
27         {
28                 print_info (argv[1]);
```

wav_info.c

This causes a crash in print_info()

Fuzzing binutils

```
input0@zero:~/fuzz101$ git clone git://sourceware.org/git/binutils-gdb.git
Cloning into 'binutils-gdb'...
remote: Counting objects: 934850, done.
remote: Compressing objects: 100% (155424/155424), done.
remote: Total 934850 (delta 768587), reused 900244 (delta 767099)
Receiving objects: 100% (934850/934850), 328.05 MiB | 694.00 KiB/s, done.
Resolving deltas: 100% (768587/768587), done.
Checking out files: 100% (32950/32950), done.
input0@zero:~/fuzz101$
```

git clone binutils

```
input0@zero:~/fuzz101/binutils-gdb$ CC=afl-clang-fast CXX=afl-clang-fast++ ./configure --disable-shared
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking target system type... x86_64-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether ln works... yes
checking whether ln -s works... yes
checking for a sed that does not truncate output... /bin/sed
checking for gawk... gawk
checking for gcc... afl-clang-fast
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether afl-clang-fast accepts -g... yes
checking for afl-clang-fast option to accept ISO C89... none needed
checking whether we are using the GNU C++ compiler... yes
checking whether afl-clang-fast++ accepts -g... yes
checking whether g++ accepts -static-libstdc++ -static-libgcc... no
checking for gnatbind... no
checking for gnatmake... no
checking whether compiler driver understands Ada... no
checking how to compare bootstrapped objects... cmp --ignore-initial=16 $$f1 $$f2
checking for objdir... .libs
checking for isl 0.15 or later... no
required isl version is 0.15 or later
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 1 locations (non-hardened mode, ratio 100%).
checking for default BUILD_CONFIG...
checking for --enable-vtable-verify... no
```

Using cross compilers by setting CC and CXX flags so that GNU calls CC rather than native complier

In this case we have used AFLC, so the binary is instrumented by AFLC

Fuzzing binutils

```
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 58 locations (non-hardened mode, ratio 100%).
  CXX     amd64-linux-nat.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 93 locations (non-hardened mode, ratio 100%).
  CXX     amd64-linux-tdep.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 375 locations (non-hardened mode, ratio 100%).
  CXX     amd64-nat.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 86 locations (non-hardened mode, ratio 100%).
  CXX     amd64-tdep.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 749 locations (non-hardened mode, ratio 100%).
  CXX     annotate.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 294 locations (non-hardened mode, ratio 100%).
  CXX     arch-utils.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 207 locations (non-hardened mode, ratio 100%).
  CXX     arch/amd64.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 18 locations (non-hardened mode, ratio 100%).
  CXX     arch/i386.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 19 locations (non-hardened mode, ratio 100%).
  CXX     auto-load.o
afl-clang-fast 2.52b by <lszekeres@google.com>
afl-llvm-pass 2.52b by <lszekeres@google.com>
[+] Instrumented 1003 locations (non-hardened mode, ratio 100%).
```

Understanding the inputs taken by readelf

```
input0@zero:~/fuzz101/binutils-gdb/binutils$ ./readelf -h /bin/ps
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0xb3b0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          131640 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         28
  Section header string table index: 27
input0@zero:~/fuzz101/binutils-gdb/binutils$
```

Once binary is configured then `make`

Fuzzing binutils

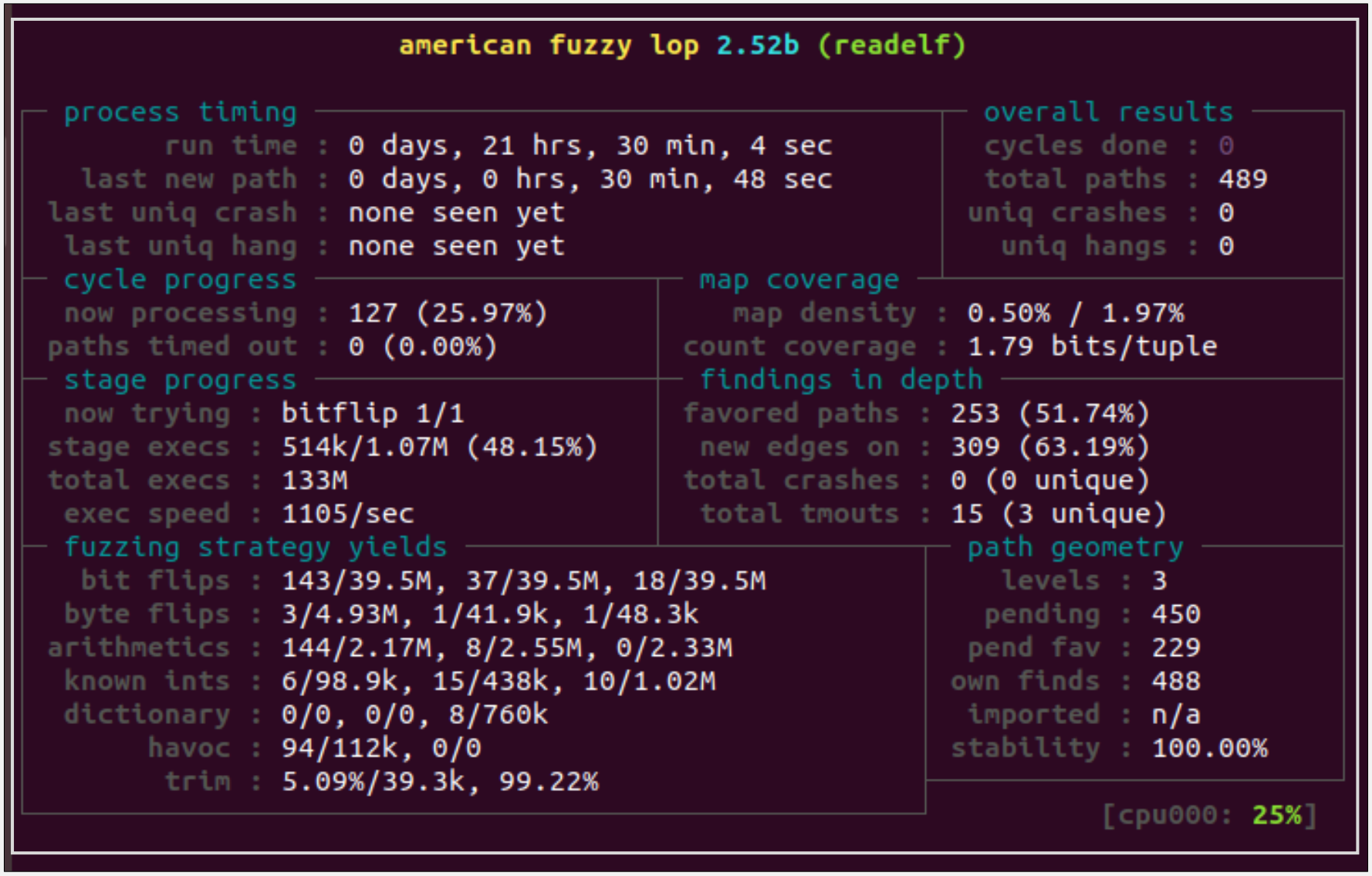
```
input0@zero:~/fuzz101/binutils-gdb/binutils$ mkdir in
input0@zero:~/fuzz101/binutils-gdb/binutils$ mkdir out
input0@zero:~/fuzz101/binutils-gdb/binutils$ cp /bin/ps in/
input0@zero:~/fuzz101/binutils-gdb/binutils$
```

Creating input/output directory

```
input0@zero:~/fuzz101/binutils-gdb/binutils$ afl-fuzz -i in/ -o out/ ./readelf -h @@
afl-fuzz 2.52b by <lcamtuf@google.com>
[+] You have 4 CPU cores and 1 runnable tasks (utilization: 25%).
[+] Try parallel jobs - see /usr/share/doc/afl-doc/docs/parallel_fuzzing.txt.
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Checking CPU scaling governor...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning 'in/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:ps'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 133432, map size = 379, exec speed = 595 us
[+] All test cases processed.

[!] WARNING: Some test cases are huge (130 kB) - see /usr/share/doc/afl-doc/docs/perf_tips.txt!
[+] Here are some useful stats:

    Test case count : 1 favored, 0 variable, 1 total
       Bitmap range : 379 to 379 bits (average: 379.00 bits)
        Exec timing : 595 to 595 us (average: 595 us)

[*] No -t option specified, so I'll use exec timeout of 20 ms.
[+] All set and ready to roll!
```

Fuzzing using afl-fuzz with the attribute –h of readelf

Fuzzing binutils

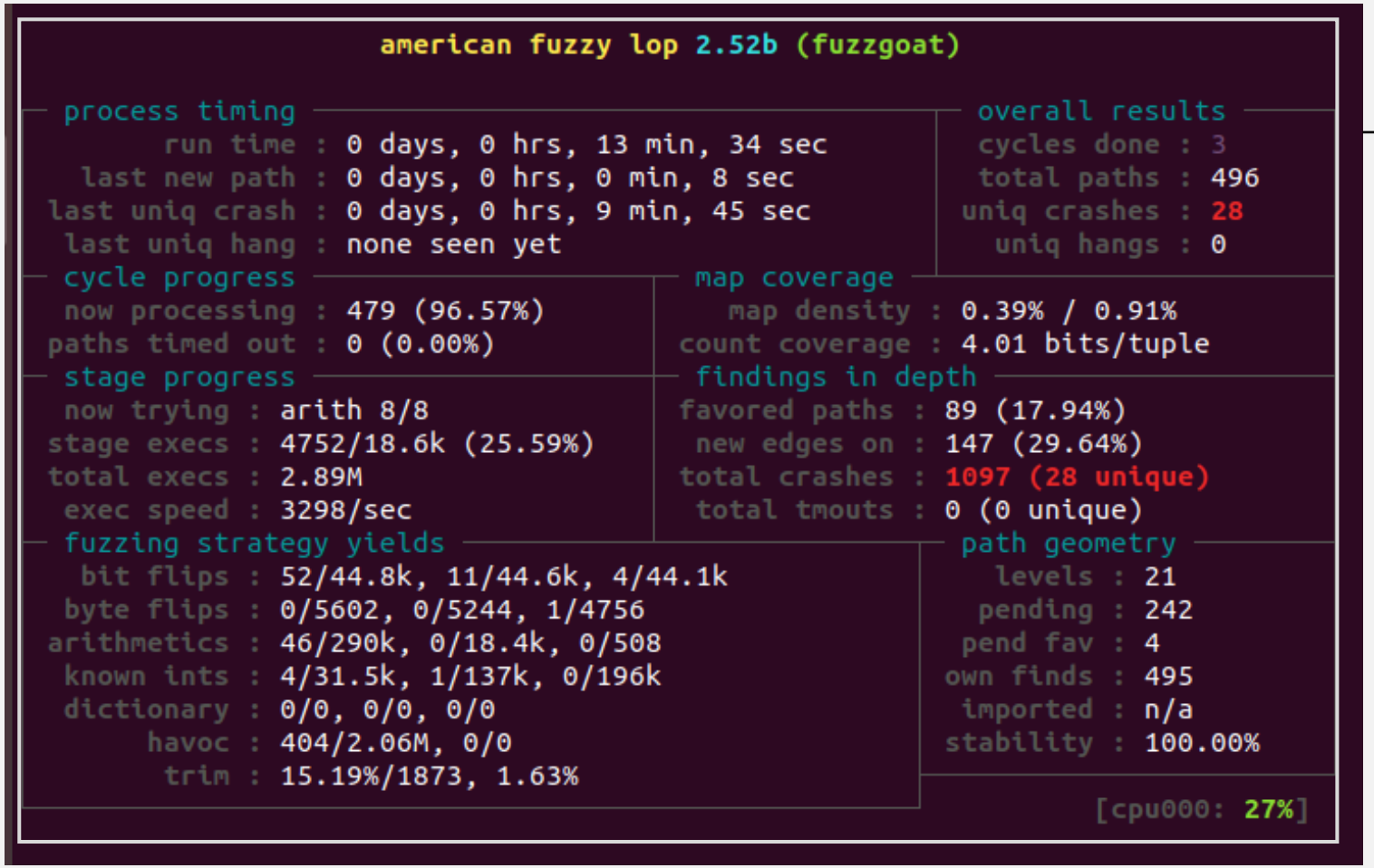


More Demo!

Fuzzing pngwriter & libpng

https://github.com/pngwriter/pngwriter

FYI - FuzzGoat



FuzzGoat - A vulnerable C program for testing fuzzer's but the crashes which gets generated while fuzzing FuzzGoat with AFL would be a good exercise to debug.

Targets to fuzz

Targets		
Libpng	PHP	Libtiff
OpenSSL	PuTTY	Nginx
Apache	Ffmpeg	OpenSSH
Dpkg	Rcs	Libyaml
Glibc	Perl	Python
Radare2	Curl	Exiv
Libbpg	Libraw	Libbson
Libsass	Openjpg	MySQL
Mpg123	Zstd	Webkit
Lrzip	Gnulib	Ettercap

and many more …

Still need more targets?

```
input0@zero:~/fuzz101$ apt-get source libwe
libweasel-perl                      libwebkitgtk-1.0-0                  libwebservice-validator-html-w3c-perl
libweather-ion7                     libwebkitgtk-3.0-0                  libwebservice-youtube-perl
libweb-api-perl                     libwebkitgtk-3.0-dev                libweb-simple-perl
libwebauth12                        libwebkitgtk-dev                    libwebsocket-api-java
libwebauth-dev                      libwebkitgtk-doc                    libwebsocketpp-dev
libwebauth-perl                     libweb-machine-perl                 libwebsocketpp-doc
libwebcam                           libweb-mrest-cli-perl               libwebsockets
libwebcam0                          libweb-mrest-perl                   libwebsockets8
libwebcam0-dbg                      libwebp                             libwebsockets-dev
libwebcam0-dev                      libwebp6                            libwebsockets-test-server
libweb-id-perl                      libwebpdemux2                       libwebsockets-test-server-common
libwebinject-perl                   libwebp-dev                         libweed0
libwebjars-locator-core-java        libwebpmux3                         libweed-dev
libwebjars-locator-java             libweb-query-perl                   libwerken.xpath-java
libwebkdc-perl                      libwebrtc-audio-processing1         libwerken.xpath-java-doc
libwebkit2gtk-4.0-37                libwebrtc-audio-processing-dev      libweston-3-0
libwebkit2gtk-4.0-37-gtk2           libweb-scraper-perl                 libweston-3-dev
libwebkit2gtk-4.0-dev               libwebservice-cia-perl              libweupnp-java
libwebkit2gtk-4.0-doc               libwebservice-musicbrainz-perl      libweupnp-java-doc
libwebkit2-sharp-4.0-cil            libwebservice-solr-perl
libwebkit2-sharp-4.0-cil-dev        libwebservice-validator-css-w3c-perl
input0@zero:~/fuzz101$ apt-get source libwebp
Reading package lists... Done
Need to get 893 kB of source archives.
Get:1 http://archive.ubuntu.com/ubuntu trusty/main libwebp 0.4.0-4 (dsc) [1,398 B]
Get:2 http://archive.ubuntu.com/ubuntu trusty/main libwebp 0.4.0-4 (tar) [888 kB]
Get:3 http://archive.ubuntu.com/ubuntu trusty/main libwebp 0.4.0-4 (diff) [3,782 B]
Fetched 893 kB in 2s (496 kB/s)
dpkg-source: info: extracting libwebp in libwebp-0.4.0
dpkg-source: info: unpacking libwebp_0.4.0.orig.tar.gz
dpkg-source: info: unpacking libwebp_0.4.0-4.debian.tar.gz
input0@zero:~/fuzz101$
```

Use
`apt-get source lib<tab><tab>`
Too many packages to fuzz.

Pro tip – Defend against dependences

```
input0@zero:~$ apt-cache search libgtk
gnome-accessibility-themes - High Contrast GTK+ 2 theme and icons
gnome-themes-extra - Adwaita GTK+ 2 theme – engine
gnome-themes-extra-data - Adwaita GTK+ 2 theme – common files
libgdk-pixbuf2.0-doc - GDK Pixbuf library (documentation)
libgtk-3-0 - GTK+ graphical user interface library
libgtk-3-bin - programs for the GTK+ graphical user interface library
libgtk-3-common - common files for the GTK+ graphical user interface library
libgtk-3-dev - development files for the GTK+ library
libgtk-3-doc - documentation for the GTK+ graphical user interface library
libgtk2.0-0 - GTK+ graphical user interface library
libgtk2.0-bin - programs for the GTK+ graphical user interface library
libgtk2.0-common - common files for the GTK+ graphical user interface library
libgtk2.0-dev - development files for the GTK+ library
libgtk2.0-doc - documentation for the GTK+ graphical user interface library
libgtk3-perl - Perl bindings for the GTK+ graphical user interface library
libgtkmm-2.4-1v5 - C++ wrappers for GTK+ (shared libraries)
libgtkmm-2.4-dev - C++ wrappers for GTK+ (development files)
libgtkmm-2.4-doc - C++ wrappers for GTK+ (documentation)
libgtkmm-3.0-1v5 - C++ wrappers for GTK+ (shared libraries)
libgtkmm-3.0-dev - C++ wrappers for GTK+ (development files)
libgtkmm-3.0-doc - C++ wrappers for GTK+ (documentation)
libgtksourceview-3.0-1 - shared libraries for the GTK+ syntax highlighting widget
libgtksourceview-3.0-common - common files for the GTK+ syntax highlighting widget
libgtksourceview-3.0-dev - development files for the GTK+ syntax highlighting widget
libgtksourceview-3.0-doc - documentation for the GTK+ syntax highlighting widget
gnome-themes-standard-data - Adwaita GTK+ 2 theme – common files
gtk2-ex-formfactory-perl - Makes building complex GUI's easy (dummy package)
libgtk-dotnet3.0-cil - GTK.NET library
libgtk-dotnet3.0-cil-dev - GTK.NET library - development files
libgtk-sharp-beans-cil - Supplementary CLI bindings for GTK 2.14+
libgtk-sharp-beans2.0-cil-dev - Supplementary CLI bindings for GTK 2.14+ - development package
libgtk-vnc-1.0-0 - VNC viewer widget for GTK+2 (runtime libraries)
libgtk-vnc-1.0-dev - VNC viewer widget for GTK+2 (development files)
libgtk-vnc-2.0-0 - VNC viewer widget for GTK+3 (runtime libraries)
libgtk-vnc-2.0-dev - VNC viewer widget for GTK+3 (development files)
libgtk2-ex-entry-pango-perl - Gtk2::Entry that accepts pango markup
libgtk2-ex-formfactory-perl - Makes building complex GUI's easy
```

Use
`apt-cache search <packagename>`

Quick Recap

101 - ASAN

Sanitizers -fsanitize=address

Maintain by google

AddressSanitizer - Detects addressability issues
MemorySanitizer - Detects use of uninitialized memory

Sanitizers -fsanitize=address

AddressSanitizer aka ASAN is a memory error detector.
Bugs, which often don't gets crash can be found by sanitizers.

- User after free – Memory corruption flaw
- Heap buffer overflow – Overflow that happens in heap data area
- Global buffer overflow – Buffers in global memory
- Stack buffer overflow – Stack smashing
- Use after return – Stack object is used after the function
- User after scope - Stack object is used outside the scope defined
- Initialization order bugs – Common issues

Sanitizers -fsanitize=memory

MemorySanitizer aka MSAN is a detector of uninitialized memory

Alternative - Valgrind (Memcheck tool)

Sanitizers -fsanitize=address

```
input0@zero:~/fuzz101$ cat bof.c
#include <stdio.h>

void fuzzing101()
{
    printf("PhDays\n");
}

void echo()
{
    char buffer[20];

    printf("Enter some text:\n");
    scanf("%s", buffer);
    printf("You entered: %s\n", buffer);
}

int main()
{
    echo();

    return 0;
}

input0@zero:~/fuzz101$ clang -fsanitize=address -O1 -fno-omit-frame-pointer -g bof.c
input0@zero:~/fuzz101$
```

bof.c is vulnerable to stack based buffer overflow.

We have used clang + ASAN to compile the C code.
-fno-omit-frame-pointer = To get stack traces.

Sanitizers -fsanitize=address

```
$ ./a.out
Enter some text:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
================================================================
==12928==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffc146dc614 at pc 0x00000043c903 bp 0x7ffc146dc4a0 sp 0x7ffc146dbc10
WRITE of size 49 at 0x7ffc146dc614 thread T0
    #0 0x43c902 in scanf_common(void*, int, bool, char const*, __va_list_tag*) /tmp/final/llvm.src/projects/compiler-
rt/lib/asan/../sanitizer_common/sanitizer_common_interceptors_format.inc:343:5
    #1 0x43d1eb in __interceptor___isoc99_vscanf /tmp/final/llvm.src/projects/compiler-
rt/lib/asan/../sanitizer_common/sanitizer_common_interceptors.inc:1415:1
    #2 0x43d1eb in __isoc99_scanf /tmp/final/llvm.src/projects/compiler-rt/lib/asan/../sanitizer_common/sanitizer_common_interceptors.inc:1436
    #3 0x4e71f9 in echo /home/input0/fuzz101/bof.c:13:5
    #4 0x4e7278 in main /home/input0/fuzz101/bof.c:19:5
    #5 0x7eff86966b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #6 0x419cf9 in _start (/home/input0/fuzz101/a.out+0x419cf9)

Address 0x7ffc146dc614 is located in stack of thread T0 at offset 52 in frame
    #0 0x4e714f in echo /home/input0/fuzz101/bof.c:9

  This frame has 1 object(s):
    [32, 52) 'buffer' (line 10) <== Memory access at offset 52 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /tmp/final/llvm.src/projects/compiler-
rt/lib/asan/../sanitizer_common/sanitizer_common_interceptors_format.inc:343:5 in scanf_common(void*, int, bool, char const*, __va_list_tag*)
Shadow bytes around the buggy address:
  0x1000028d3870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d3880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d3890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d38a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d38b0: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
=>0x1000028d38c0: 00 00[04]f3 f3 f3 f3 f3 00 00 00 00 00 00 00 00
  0x1000028d38d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d38e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d38f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d3900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000028d3910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==12928==ABORTING
```

The generated ./a.out, while running ASAN provides the feedback that the compiled binary is vulnerable to stack buffer overflow

Demo -fsanitize=address

```
input0@zero:~/Downloads/libiec61850-1.3.3$ ls
CHANGELOG  CMakeLists.txt  config  COPYING  demos  dotnet  examples  hal  make  Makefile  pyiec61850  README.md  src  third_party  tools
input0@zero:~/Downloads/libiec61850-1.3.3$ mkdir build ; cd build
input0@zero:~/Downloads/libiec61850-1.3.3/build$ cmake .. -DCMAKE_CXX_FLAGS="-fsanitize=address  -fsanitize=leak -g" -DCMAKE_C_FLAGS="-fsaniti
ze=address  -fsanitize=leak -g"
-- The C compiler identification is GNU 7.4.0
-- The CXX compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for clock_gettime in rt
-- Looking for clock_gettime in rt - found
-- Check if the system is big endian
-- Searching 16 bit integer
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for stddef.h
-- Looking for stddef.h - found
-- Check size of unsigned short
-- Check size of unsigned short - done
-- Using unsigned short
-- Check if the system is big endian - little endian
server-example-logging: sqlite not found
-- Performing Test COMPILER_HAS_HIDDEN_VISIBILITY
-- Performing Test COMPILER_HAS_HIDDEN_VISIBILITY - Success
```

Compile using cmake using `-fsanitize=`

Demo -fsanitize=address

```
input0@zero:~/Downloads/libiec61850-1.3.3/build$ cd examples/goose_publisher/
input0@zero:~/Downloads/libiec61850-1.3.3/build/examples/goose_publisher$ ./goose_publisher_example
Using interface eth0
Error creating raw socket!
ASAN:DEADLYSIGNAL
=================================================================
==1501==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x5616c67dc02b bp 0x7ffd11573180 sp 0x7ffd11573160 T0)
==1501==The signal is caused by a READ memory access.
==1501==Hint: address points to the zero page.
    #0 0x5616c67dc02a in Ethernet_sendPacket /home/input0/Downloads/libiec61850-1.3.3/hal/ethernet/linux/ethernet_linux.c:230
    #1 0x5616c67d02b1 in GoosePublisher_publish /home/input0/Downloads/libiec61850-1.3.3/src/goose/goose_publisher.c:381
    #2 0x5616c67bd533 in main /home/input0/Downloads/libiec61850-1.3.3/examples/goose_publisher/goose_publisher_example.c:63
    #3 0x7f2651f76b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #4 0x5616c67bd039 in _start (/home/input0/Downloads/libiec61850-1.3.3/build/examples/goose_publisher/goose_publisher_example+0x6039)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /home/input0/Downloads/libiec61850-1.3.3/hal/ethernet/linux/ethernet_linux.c:230 in Ethernet_sendPacket
==1501==ABORTING
input0@zero:~/Downloads/libiec61850-1.3.3/build/examples/goose_publisher$
```


cmake .. –DCMAKE_CXX_FLAGS=“-fsanitize=address –fsanitize=leak –g” –DCMAKE_C_FLAGS=“-fsanitize=address –fsanitize=leak -g”

Demo -fsanitize=address

Compiling libwav, pngwriter, FuzzGoat & libiec61850 and understanding the traces.

MSAN -fsanitize=memory

FYI, Memory leak might be false positive sometimes, would suggest cross checking with valgrind or attaching the process to gdb-server.

FYI - llvm-symbolizer

Helps you to find symbols, filenames and line numbers.

Example: https://llvm.org/docs/CommandGuide/llvm-symbolizer.html

FYI – Shared libraries

Executable (ASAN) + Library (ASAN) = Works
Executable (ASAN) + Library (NoASAN) = Works
Executable (NoASAN) + Library (ASAN) = Doesn't Work

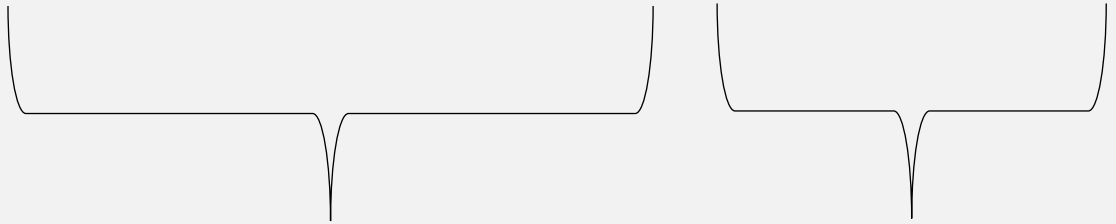
Quick Recap

101 – AFL + ASAN

AFL + ASAN = Superpower

AFL + ASAN

```
$ CC=afl-clang-fast CXX=afl-clang-fast++ ASAN_OPTIONS=symbolize=0 AFL_USE_ASAN=1
```



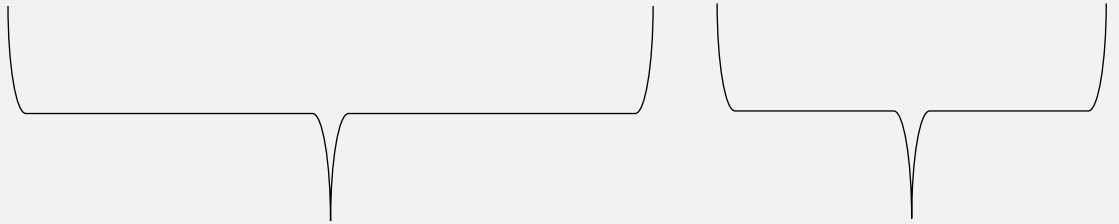
Resolving address to source code line number and filenames

Compile with ASAN

Note: AddressSanitizer will be not be compatible if `-static` is used

AFL + ASAN

```
$ CC=afl-clang-fast CXX=afl-clang-fast++ ASAN_OPTIONS=symbolize=0 AFL_USE_MSAN=1
```



Resolving address to source code line number and filenames

Compile with MSAN

101 – Protocol Fuzzing

Fuzzing network protocols

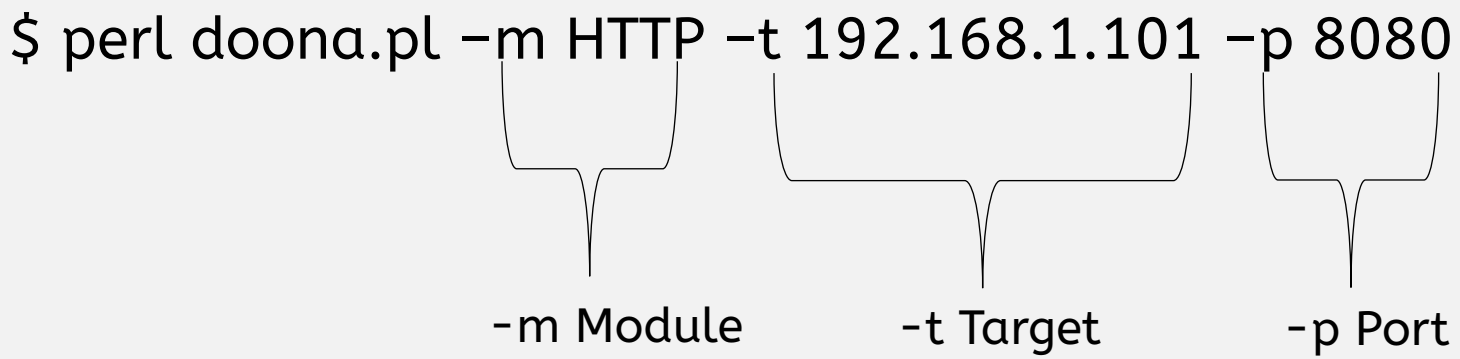
We have used doona and boofuzz to fuzz network based protocol such as HTTP, FTP, SMTP etc.

- doona - Network fuzzing tool
  https://github.com/wireghoul/doona

- boofuzz - Network fuzzing tool
  pip install boofuzz

Fuzzing network protocols

- doona - Network fuzzing tool
  https://github.com/wireghoul/doona



How doona works?

```
my @modules = map { s!bedmod/(.*)\.pm!$1!; $_ } glob("bedmod/*.pm");

# the hope is to overwrite a return pointer on the stack,
# making the server execute invalid code and crash
my @overflowstrings = (
    "A" x 33, "A" x 254, "A" x 255, "A" x 256, "A" x 257, "A" x 1023, "A" x 1024, "A" x 1025, "A" x 1026,
    "A" x 1044, "A" x 2047, "A" x 2048, "A" x 2049, "A" x 2068, "A" x 3092, "A" x 4116, "A" x 5140,
    "A" x 6164, "A" x 7188, "A" x 8212, "A" x 9236, "A" x 10260, "A" x 11284, "A" x 12308, "A" x 13332,
    "A" x 14356, "A" x 15380,
    "\\" x 200, "\\" x 255, "\\" x 256, "\\" x 9000,
    "/" x 200, "/" x 255, "/" x 256, "/" x 9000,
    "A/" x 256, "AA/" x 256, "AAA/" x 256, "AAAA/" x 256,
    "." x 200, "." x 255, "." x 256, "." x 9000, " " x 9000, "AA " x 200,
);
my @formatstrings = (
    "%s" x 4, "%s%p%x%d", "%s" x 8, "%s" x 15, "%s" x 30, "%.1024d", "%.2048d", "%.4096d", '%@' x 53, "%.16i705u%2\$hn", "%#123456x"
);

# three ansi overflows, two ansi format strings, two OEM Format Strings
my @unicodestrings = ("\x99" x 4, "\x99" x 512, "\x99" x 1024, "\xCD" x 10, "\xCD" x 40, "\xCB" x 10, "\xCB"x40);
my @largenumbers = (
    "255", "256", "257",
    "65535", "65536", "65537",
    "16777215", "16777216", "16777217",
    "2147483647", "2147483648", "2147483649",
    "0xfffffffe", "0xffffffff", "4294967295",
    "9223372036854775807", "18446744073709551615",
    "0", "-1", "-268435455", "-20",
    "2.2250738585072011e-308",
  );
my @miscstrings = (
    "/", "\\", "%0xa", " ", "+", "<", ">", "<>",
    "%", "-", "+", "*", ".", ":", ";", "&", "%u000",
    "%xx", "\\x41", "%00", "\x00", "\x01\x01\x01\x01",
    "A\@A.COM","AAAA.ABCD","AAAA://AAAAA.AAAAA/AAAA",
    "\t", "\r", "\r\n", "\n"
);
my $idx = 0;
my $prevfuzz = '';
# print "\n Doona $VERSION by Wireghoul (www.justanotherhacker.com)\n\n";

# get the parameters we need for every test
getopts('m:s:t:o:p:r:u:v:w:x:M:c:dhk');
$opt_s = $opt_m if ($opt_m);
&usage unless($opt_s);
$opt_s = lc($opt_s);                        # convert it to lowercase
```

Overflow strings
Format strings
Unicode strings
….

Example

```
zero@ubuntu:~$ python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
172.16.4.1 - - [05/May/2019 01:08:09] "HEAD / HTTP/1.0" 200 -
172.16.4.1 - - [05/May/2019 01:08:09] code 400, message Bad request syntax ('AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA')
172.16.4.1 - - [05/May/2019 01:08:09] "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" 400 -
172.16.4.1 - - [05/May/2019 01:08:11] code 400, message Bad request syntax ('AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA')
172.16.4.1 - - [05/May/2019 01:08:11] "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" 400 -
172.16.4.1 - - [05/May/2019 01:08:13] code 400, message Bad request syntax ('AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA')
172.16.4.1 - - [05/May/2019 01:08:13] "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" 400 -
172.16.4.1 - - [05/May/2019 01:08:15] code 400, message Bad request syntax ('AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA')
```

```
Dhiraj@Dhirajs-MacBook-Pro:~/Desktop/doona$ perl doona.pl -m HTTP -t 172.16.4.210 -p 8080
 + Buffer overflow testing
    1/39    [XAXAX] ............................................. (45)
    2/39    [XAXAX / HTTP/1.0] ............................................. (90)
    3/39    [HEAD XAXAX HTTP/1.0] ............................................. (135)
    4/39    [HEAD /XAXAX HTTP/1.0] ............................................. (180)
    5/39    [HEAD /?XAXAX HTTP/1.0] ............................................. (225)
    6/39    [HEAD / XAXAX] ...
```

Fuzzing **SimpleHTTPServer** module of Python

Fuzzing Mozilla PDF.js

It was observed that the development server used in PDF.js gets crash when a malformed URI(bad request) is sent.

```
git clone https://github.com/mozilla/pdf.js.git
cd pdf.js
npm install -g gulp-cli
npm install
gulp server
```

```
Server running at http://localhost:8888/
[12:16:22] 'server' errored after 1.01 h
[12:16:22] URIError: URI malformed
at decodeURI (<anonymous>)
at WebServer._handler (/Users/Dhiraj/Desktop/pdf.js/test/webserver.js:86:35)
at Server.emit (events.js:188:13)
at Server.EventEmitter.emit (domain.js:459:23)
at parserOnIncoming (_http_server.js:676:12)
at HTTPParser.parserOnHeadersComplete (_http_common.js:113:17)
```

Understanding the crash – PDF.js

```
   _handler: function (req, res) {
     var url = req.url.replace(/\/\//g, '/');
     var urlParts = /([^?]*)((?:\?(.*))?)/.exec(url);
-    // guard against directory traversal attacks,
-    // e.g. /../../../../../../../etc/passwd
-    // which let you make GET requests for files outside of this.root
-    var pathPart = path.normalize(decodeURI(urlParts[1]));
     var queryPart = urlParts[3];
     var verbose = this.verbose;
```

```
80     _handler: function (req, res) {
81       var url = req.url.replace(/\/\//g, '/');
82       var urlParts = /([^?]*)((?:\?(.*))?)/.exec(url);
83   +   try {
84   +     // Guard against directory traversal attacks such as
85   +     // `/../../../../../../../etc/passwd`, which let you make GET requests
86   +     // for files outside of `this.root`.
87   +     var pathPart = path.normalize(decodeURI(urlParts[1]));
88   +   } catch (ex) {
89   +     // If the URI cannot be decoded, a `URIError` is thrown. This happens for
90   +     // malformed URIs such as `http://localhost:8888/%s%s` and should be
91   +     // handled as a bad request.
92   +     res.writeHead(400);
93   +     res.end('Bad request', 'utf8');
94   +     return;
95   +   }
96       var queryPart = urlParts[3];
97       var verbose = this.verbose;
```

https://github.com/lpillonel/pdf.js/commit/df65c03e12f2878bff9ff44247f383c016bcc206

Fuzzing with boofuzz - https://www.inputzero.io/2019/01/fuzzing-http-servers.html

Gain reputation on H1

Then fuzz NodeJS modules (https://hackerone.com/nodejs-ecosystem)

- http-server
- simple-server
- Serve
- local-web-server
- simplehttpserver

101 – Our Research

CVE-2019-8375 - WebKit

The UIProcess subsystem in WebKit does not prevent the script dialog size from exceeding the web view size

```
(epiphany:4423): Gdk-WARNING **: Native Windows wider or taller than 32767 pixels are
not supported
Gdk-Message: Error 71 (Protocol error) dispatching to Wayland display
```

```
<script>
   var a = '';
   for (var i = 1; i <= 5000; i++)
    {
       a += 'A';
    }
   alert(a);
</script>
```

Reference: https://www.inputzero.io/2019/02/fuzzing-webkit.html

CVE-2019-8375 - WebKit

The UIProcess subsystem in WebKit does not prevent the script dialog size from exceeding the web view size

```
static GtkWidget* webkitWebViewCreateJavaScriptDialog(WebKitWebView* webView,
GtkMessageType type, GtkButtonsType buttons, int defaultResponse, const char*
primaryText, const char* secondaryText = nullptr)
{
    GtkWidget* parent = gtk_widget_get_toplevel(GTK_WIDGET(webView));
    GtkWidget* dialog =
gtk_message_dialog_new(WebCore::widgetIsOnscreenToplevelWindow(parent) ?
GTK_WINDOW(parent) : nullptr,
        GTK_DIALOG_DESTROY_WITH_PARENT, type, buttons, "%s", primaryText);
    if (secondaryText)
        gtk_message_dialog_format_secondary_text(GTK_MESSAGE_DIALOG(dialog), "%s",
secondaryText);
    GUniquePtr<char> title(g_strdup_printf("JavaScript - %s",
webkitWebViewGetPage(webView).pageLoadState().url().utf8().data()));
    gtk_window_set_title(GTK_WINDOW(dialog), title.get());
    if (buttons != GTK_BUTTONS_NONE)
        gtk_dialog_set_default_response(GTK_DIALOG(dialog), defaultResponse);

    return dialog;
}
```

Reference: https://www.inputzero.io/2019/02/fuzzing-webkit.html

CVE-2019-6439 - WolfSSL

A heap-based buffer overflow - It was a possible heapbuffer overflow when using 16KB test packets.

```
==4088==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x619000000480 at pc 0x00000050ff16 bp 0x7fef206fdbf0 sp 0x7fef206fdbe8
WRITE of size 1 at 0x619000000480 thread T2
    #0 0x50ff15  (/wolfssl/examples/benchmark/tls_bench+0x50ff15)
    #1 0x4dfa52  (/wolfssl/examples/benchmark/tls_bench+0x4dfa52)
    #2 0x7fef243ac6da  (/lib/x86_64-linux-gnu/libpthread.so.0+0x76da)
    #3 0x7fef23ab188e  (/lib/x86_64-linux-gnu/libc.so.6+0x12188e)

0x619000000480 is located 0 bytes to the right of 1024-byte region [0x619000000080,0x619000000480)
allocated by thread T2 here:
    #0 0x4d1fa0  (/wolfssl/examples/benchmark/tls_bench+0x4d1fa0)
    #1 0x50f277  (/wolfssl/examples/benchmark/tls_bench+0x50f277)
    #2 0x4dfa52  (/wolfssl/examples/benchmark/tls_bench+0x4dfa52)

Thread T2 created by T0 here:
    #0 0x435490  (/wolfssl/examples/benchmark/tls_bench+0x435490)
    #1 0x50cbf5  (/wolfssl/examples/benchmark/tls_bench+0x50cbf5)
    #2 0x5101d0  (/wolfssl/examples/benchmark/tls_bench+0x5101d0)
    #3 0x7fef239b1b96  (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: heap-buffer-overflow (/wolfssl/examples/benchmark/tls_bench+0x50ff15)
Shadow bytes around the buggy address:
  0x0c327fff8040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c327fff8050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c327fff8060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c327fff8070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c327fff8080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c327fff8090:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c327fff80a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c327fff80b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c327fff80c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c327fff80d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c327fff80e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==4088==ABORTING
```

```
/* Allocate and initialize a packet sized buffer */
writeBuf = (unsigned char*)XMALLOC(info->packetSize, NULL,
    DYNAMIC_TYPE_TMP_BUFFER);
if (writeBuf != NULL) {
    XMEMSET(writeBuf, 0, info->packetSize);
    XSTRNCPY((char*)writeBuf, kTestStr, info->packetSize);
}
```

Reference: https://github.com/wolfSSL/wolfssl/pull/2013

phd Positive Hack Days

CVE-2019-6498 - gattlib

A stack-based buffer overflow

```
==31499==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffc99cec2d4 at pc 0x00000044de04 bp 0x7ffc99cec270 sp 0x7ffc99ceba20
READ of size 21 at 0x7ffc99cec2d4 thread T0
    #0 0x44de03 in __interceptor_strlen.part.30 (/home/zero/gattlib/build/examples/discover/discover+0x44de03)
    #1 0x7f149e22069e in gattlib_connect /home/zero/gattlib/dbus/gattlib.c:224:18
    #2 0x50bf48 in main /home/zero/gattlib/examples/discover/discover.c:43:15
    #3 0x7f149c6d6b96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c:310
    #4 0x41c959 in _start (/home/zero/gattlib/build/examples/discover/discover+0x41c959)

Address 0x7ffc99cec2d4 is located in stack of thread T0 at offset 84 in frame
    #0 0x7f149e22056f in gattlib_connect /home/zero/gattlib/dbus/gattlib.c:209

  This frame has 3 object(s):
    [32, 40) 'error'
    [64, 84) 'device_address_str' <== Memory access at offset 84 overflows this variable
    [128, 228) 'object_path'
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/home/zero/gattlib/build/examples/discover/discover+0x44de03) in __interceptor_strlen.part.30
Shadow bytes around the buggy address:
  0x100013395800: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100013395810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100013395820: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100013395830: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100013395840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100013395850: f1 f1 f1 f1 00 f2 f2 f2 00 00[04]f2 f2 f2 f2 f2
  0x100013395860: 00 00 00 00 00 00 00 00 00 00 00 00 04 f3 f3 f3
  0x100013395870: f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00 00 00
  0x100013395880: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
  0x100013395890: 00 f2 f2 f2 00 f2 f2 f2 04 f2 04 f2 00 00 00 00
  0x1000133958a0: 06 f3 f3 f3 f3 f3 f3 f3 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==31499==ABORTING
```

```
./discover `python -c 'print "A"*20'`
```

```c
        if (argc != 2) {
                printf("%s <device_address>\n", argv[0]);
                return 1;
        }

        connection = gattlib_connect(NULL, argv[1], BDADDR_LE_PUBLIC, BT_SEC_LOW, 0, 0);
        if (connection == NULL) {
                fprintf(stderr, "Fail to connect to the bluetooth device.\n");
                return 1;
}
```

Reference: https://github.com/labapart/gattlib/issues/81

CVE-2018-18957 - libiec61850

A stack-based buffer overflow, misusing strncpy

```
(gdb) run crash_goosecr_stack_smash_overflow_aaaaaaaaa
Starting program: /home/input0/Desktop/libiec61850/examples/goose_publisher/goose_publisher_example
crash_goosecr_stack_smash_overflow_aaaaaaaaa
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Using interface crash_goosecr_stack_smash_overflow_aaaaaaaaa
*** stack smashing detected ***: <unknown> terminated

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
51                              ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
(gdb) bt
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
#1  0x00007ffff7805801 in __GI_abort () at abort.c:79
#2  0x00007ffff784e897 in __libc_message (action=action@entry=do_abort, fmt=fmt@entry=0x7ffff797b988 "*** %s ***: %s terminated\n")
    at ../sysdeps/posix/libc_fatal.c:181
#3  0x00007ffff78f9cd1 in __GI___fortify_fail_abort (need_backtrace=need_backtrace@entry=false,
    msg=msg@entry=0x7ffff797b966 "stack smashing detected") at fortify_fail.c:33
#4  0x00007ffff78f9c92 in __stack_chk_fail () at stack_chk_fail.c:29
#5  0x000055555555a211 in Ethernet_getInterfaceMACAddress (interfaceId=0x7fffffffdeee "crash_goosecr_stack_smash_overflow_aaaaaaaaa",
    addr=0x7fffffffd91c "k_smas\377\377") at hal/ethernet/linux/ethernet_linux.c:170
#6  0x00005555555594ee in prepareGooseBuffer (self=0x5555557637d0, parameters=0x7fffffffd9ac,
    interfaceID=0x7fffffffdeee "crash_goosecr_stack_smash_overflow_aaaaaaaaa") at src/goose/goose_publisher.c:168
#7  0x0000555555559293 in GoosePublisher_create (parameters=0x7fffffffd9ac,
    interfaceID=0x7fffffffdeee "crash_goosecr_stack_smash_overflow_aaaaaaaaa") at src/goose/goose_publisher.c:72
#8  0x0000555555555387 in main (argc=2, argv=0x7fffffffdaa8) at goose_publisher_example.c:52
(gdb) i r
rax            0x0         0
rbx            0x7fffffffd6b0          140737488344752
rcx            0x7ffff7803e97          140737345765015
rdx            0x0         0
rsi            0x7fffffffd410          140737488344080
rdi            0x2         2
rbp            0x7fffffffd840          0x7fffffffd840
rsp            0x7fffffffd410          0x7fffffffd410
r8             0x0         0
r9             0x7fffffffd410          140737488344080
r10            0x8         8
r11            0x246       582
r12            0x7fffffffd6b0          140737488344752
r13            0x1000      4096
r14            0x0         0
r15            0x30        48
rip            0x7ffff7803e97          0x7ffff7803e97 <__GI_raise+199>
eflags         0x246       [ PF ZF IF ]
cs             0x33        51
ss             0x2b        43
ds             0x0         0
es             0x0         0
fs             0x0         0
gs             0x0         0
(gdb)
```

Reference: https://github.com/mz-automation/libiec61850/issues/83

Memory leaks – PuTTY (H1)

#481591 **Memory leak in PSFTP**

State	Triaged (Open)	Severity	Low (0.1 ~ 3.9)
Reported To	PuTTY (European Commission - DIGIT)	Participants	
Asset	https://www.chiark.greenend.org.uk/~s... (Source code)	Visibility	Private
Weakness	Improper Access Control - Generic		
Bounty	$285.34		

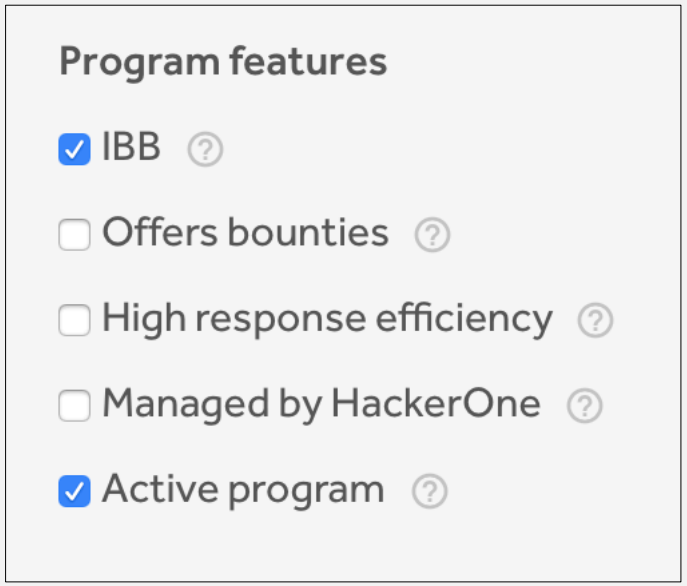
#519030 **Memory leak in PSFTP**

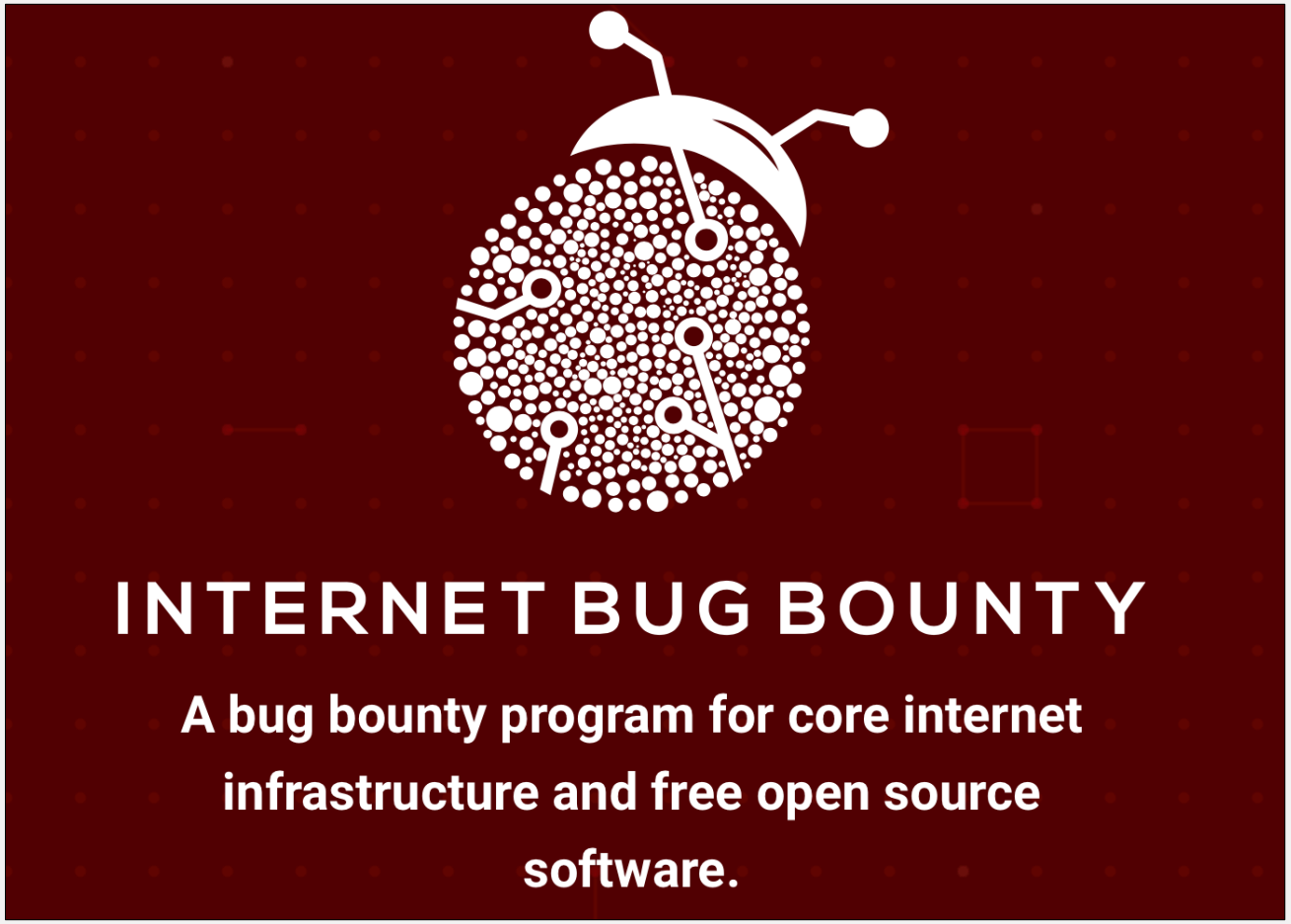
State	Resolved (Closed)	Severity	Low (0.1 ~ 3.9)
Reported To	PuTTY (European Commission - DIGIT)	Participants	
Asset	https://www.chiark.greenend.org.uk/~s... (Source code)	Visibility	Private
Weakness	Memory Corruption - Generic		
Bounty	$281.14		

101 – Fuzzing + IBB = $$$

Fuzzing + IBB = $$$

Go to HackerOne → Directory, apply the below filter.





Reference: https://internetbugbounty.org

101 – More fuzzer’s ahead

What else?

winAFL – AFL for fuzzing Windows binaries

Hongfuzz - Security oriented fuzzer

Kitty - Fuzzing framework written in python

Sulley - Unattended fuzzing framework

Clusterfuzz - Scalable fuzzing infrastructure.

and what?

Fuzzing JavaScript libraries

Octo - A fuzzing library in JavaScript

Fuzzilli - A JavaScript Engine Fuzzer

Thank you

RandomDhiraj

dhiraj@inputzero.io

p1ngfl0yd

zubin@0mail.me

Thank you & References

We would also like to thank below mention researchers for their awesome work in fuzzing.

- Botwicz and Wojciech
- Hanno
- Google P0
- Gynvael Coldwind
- Team Serious
- Hongxu Chen


"That's all Folks!"