AFL - American fuzzy lop

Unknown macro: 'html'

List

- Description
  - Site
  - Install
  - Commands
- Description of commands
  - afl-fuzz
    - White-box, Black-box test
    - Parallel fuzzing
      - Single-system parallelization
      - Multi-system parallelization
  - afl-analyze
  - afl-cmin
  - afl-tmin
  - afl-gotcpu
  - afl-plot
  - alf-whatsup
- Example
  - Example code
  - Create to Test cases.
  - White-box testing
    - Build using afl-gcc.
    - Run afl-fuzz
  - Black-box testing
    - Install library files
    - Build using gcc
    - Run afl-fuzz
  - Check for the crash.
- Related information

Description

- **AFL(American Fuzzy Lop) (Code coverage) (Genetic algorithm) fuzzer.**
  - OS Linux, OpenBSD, FreeBSD, NetBSD 32bit 64bit .
    - MacOS X Solaris .
  - C, C++, Objective C .
  - gcc, g++, clang, clang++ .
  - White-box, Black-box .
    - QEMU .

Site

- http://lcamtuf.coredump.cx/afl/

Install

```
$ wget http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz
$ tar -xvf afl-latest.tgz
$ cd afl-2.49b/
$ make
$ sudo make install
```

Commands

Command	Description	Basic methods of use
afl-analyze		afl-analyze -i <test case file> target_app
afl-clang	clang wrapper	clang .
afl-clang++	clang++ wrapper	clang++ .
afl-cmin		afl-cmin -i <test case dir> -o <output dir> target_app
afl-fuzz	AFL	afl-fuzz -i <test case dir> -o <output dir> target_app
afl-g++	g++ wrapper	g++ .
afl-gcc	gcc wrapper	gcc .
afl-gotcpu	CPU	afl-gotcpu
afl-plot	- "gnuplot"	afl-plot <afl state dir> <graph output dir>
afl-tmin		afl-tmin -i <test case file> -o <output file> target_app
afl-whatsup		afl-whatsup <afl_sync_dir>

Description of commands

afl-fuzz

- .

White-box, Black-box test

- **White-box, Black-box .**
  - White-box afl .
  - Black-box afl QEMU , -Q .
- White-box .

White-box

```
afl-fuzz -i <test case dir> -o <output dir> target_app
```

- black-box .

Black-box

```
afl-fuzz -Q -i <test case dir> -o <output dir> target_app
```

Standard input

```
afl-fuzz -i <test case dir> -o <output dir> target_app [params...]
```

File input

```
afl-fuzz -i <test case dir> -o <output dir> target_app @@
```

Parallel fuzzing

- .
  - afl-fuzz CPU .
  - ,n n
    - afl-gotcpu
  - .
  - .

Single-system parallelization

- ("sync dir") .
  - .
- (-M) .

```
./afl-fuzz -i testcase_dir -o sync_dir -M fuzzer01 [...other stuff...]
```

  - (-S) .

```
$ ./afl-fuzz -i testcase_dir -o sync_dir -S fuzzer02 [...other stuff...]
$ ./afl-fuzz -i testcase_dir -o sync_dir -S fuzzer03 [...other stuff...]
```

Multi-system parallelization

- .
- .
- <fuzzer_id> "/queue/" .

```
for s in {1..10}; do
      ssh user@host${s} "tar -czf - sync/host${s}_fuzzid*/[qf]*" >host${s}.tgz
    done
```

-

```
  for s in {1..10}; do
      for d in {1..10}; do
        test "$s" = "$d" && continue
        ssh user@host${d} 'tar -kxzf -' <host${s}.tgz
      done
    done
```

> **Parallel fuzzing using AFL**
>
> - https://raw.githubusercontent.com/mirrorer/afl/master/docs/parallel_fuzzing.txt

afl-analyze

- **Test case .**
  - , .
- .
  - no-op block
  - Critical stream
  - "magic value"
  -
  -

    -
    - Magic
- .

afl-analyze -i testcase/test1.txt ./test

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-analyze -i testcase/test1.txt ./test

afl-analyze 2.49b by <lcamtuf@google.com>


[+] Read 4 bytes from 'testcase/test1.txt'.
[*] Performing dry run (mem limit = 50 MB, timeout = 1000 ms)...
[*] Analyzing input file (this may take a while)...

     01  - no-op block              01  - suspected length field
     01  - superficial content      01  - suspected cksum or magic int
     01  - critical stream          01  - suspected checksummed block
     01  - "magic value" section

[000000]  a  #0a  a  #0a

[+] Analysis complete. Interesting bits: 0.00% of the input file.
[+] We're done here. Have a nice day!

lazenca0x0@ubuntu:~/Documents/AFL/test$
```

afl-cmin

- **Test case .**
- test case .
    - 4 test case 7 tuple , 2 .
    - test case "Create to Test cases." .

afl-cmin -i testcase/ -o newTestCase/ ./test

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-cmin -i testcase/ -o newTestCase/ ./test
corpus minimization tool for afl-fuzz by <lcamtuf@google.com>

[*] Testing the target binary...
[+] OK, 4 tuples recorded.
[*] Obtaining traces for input files in 'testcase/'...
    Processing file 4/4...
[*] Sorting trace sets (this may take a while)...
[+] Found 7 unique tuples across 4 files.
[*] Finding best candidates for each tuple...
    Processing file 4/4...
[*] Sorting candidate list (be patient)...
[*] Processing candidates and writing output files...
    Processing tuple 7/7...
[+] Narrowed down to 2 files, saved in 'newTestCase/'.

lazenca0x0@ubuntu:~/Documents/AFL/test$ cd newTestCase/
lazenca0x0@ubuntu:~/Documents/AFL/test/newTestCase$ ls -al
total 16
drwxrwxr-x 2 lazenca0x0 lazenca0x0 4096 Aug 15 20:08 .
drwxrwxr-x 5 lazenca0x0 lazenca0x0 4096 Aug 15 20:08 ..
-rw-rw-r-- 2 lazenca0x0 lazenca0x0    4 Aug  9 00:18 test1.txt
-rw-rw-r-- 2 lazenca0x0 lazenca0x0    9 Aug  9 00:19 test3.txt
lazenca0x0@ubuntu:~/Documents/AFL/test/newTestCase$
```

afl-tmin

- **Test case .**
- .
    -

    -
    -
- Test case .
    - 68 56 .
    - 0(0x30) .
    - Test case .

afl-tmin -i result/crashes/id\:000000\,sig\:11\,src\:000000\,op\:havoc\,rep\:128 -o testcaseMin ./test

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-tmin -i result/crashes/id\:000000\,sig\:11\,src\:000000\,op\:havoc\,
rep\:128 -o testcaseMin ./test
afl-tmin 2.49b by <lcamtuf@google.com>

[+] Read 68 bytes from 'result/crashes/id:000000,sig:11,src:000000,op:havoc,rep:128'.
[*] Performing dry run (mem limit = 50 MB, timeout = 1000 ms)...
[+] Program exits with a signal, minimizing in crash mode.
[*] Stage #0: One-time block normalization...
[+] Block normalization complete, 68 bytes replaced.
[*] --- Pass #1 ---
[*] Stage #1: Removing blocks of data...
    Block length = 4, remaining size = 68
    Block length = 2, remaining size = 56
    Block length = 1, remaining size = 56
[+] Block removal complete, 12 bytes deleted.
[*] Stage #2: Minimizing symbols (1 code point)...
[+] Symbol minimization finished, 0 symbols (0 bytes) replaced.
[*] Stage #3: Character minimization...
[+] Character minimization done, 0 bytes replaced.
[*] --- Pass #2 ---
[*] Stage #1: Removing blocks of data...
    Block length = 4, remaining size = 56
    Block length = 2, remaining size = 56
    Block length = 1, remaining size = 56
[+] Block removal complete, 0 bytes deleted.

     File size reduced by : 17.65% (to 56 bytes)
    Characters simplified : 121.43%
     Number of execs done : 33
          Fruitless execs : path=12 crash=0 hang=0

[*] Writing output to 'testcaseMin'...
[+] We're done here. Have a nice day!
lazenca0x0@ubuntu:~/Documents/AFL/test$ hexdump result/crashes/id\:000000\,sig\:11\,src\:000000\,op\:havoc\,
rep\:128
0000000 81b9 ad13 0000 76e1 04ff 007f eee7 ffff
0000010 64ff 0000 798a 9379 7980 7979 7966 e100
0000020 ff76 7fc0 e700 ffee ffff ffff 7f04 e700
0000030 ffee ffff 0064 6900 7979 7993 7979 0079
0000040 0100 ff00
0000044
lazenca0x0@ubuntu:~/Documents/AFL/test$ hexdump testcaseMin
0000000 3030 3030 3030 3030 3030 3030 3030 3030
*
0000038
lazenca0x0@ubuntu:~/Documents/AFL/test$ cat testcaseMin
0000000000000000000000000000000000000000000000000000000000000000
```

afl-gotcpu

- **afl-fuzz cpu .**

afl-gotcpu

```
lazenca0x0@ubuntu:~$ afl-gotcpu
afl-gotcpu 2.49b by <lcamtuf@google.com>
[*] Measuring per-core preemption rate (this will take 1.00 sec)...
    Core #0: CAUTION (231%)

>>> CAUTION: You may still have 1 core available. <<<

lazenca0x0@ubuntu:~$
```

afl-plot

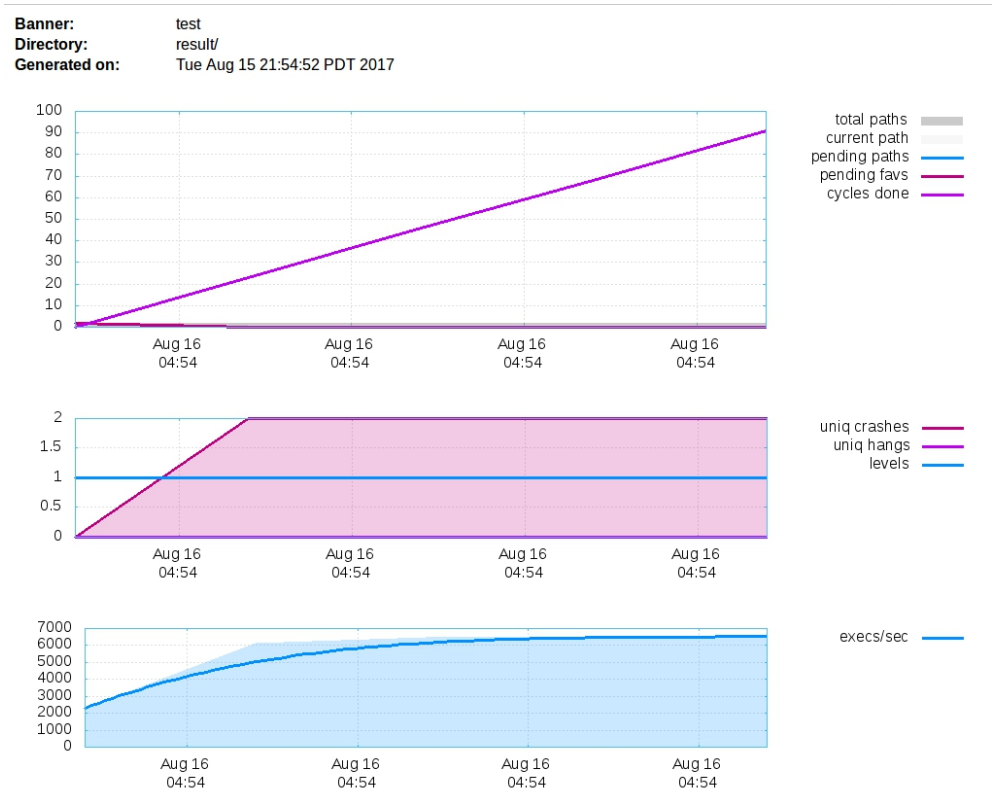
- **fuzz .**
- , index.html .

afl-plot result/ graph/

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-plot result/ graph/
progress plotting utility for afl-fuzz by <lcamtuf@google.com>

[*] Generating plots...
[*] Generating index.html...
[+] All done - enjoy your charts!
lazenca0x0@ubuntu:~/Documents/AFL/test$
```

index.html



alf-whatsup

- **fuzzer .**

afl-whatsup result/

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-whatsup result/
status check tool for afl-fuzz by <lcamtuf@google.com>

Individual fuzzers
==================

>>> fuzzer1 (0 days, 0 hrs) <<<

  cycle 1, lifetime speed 1 execs/sec, path 0/2 (0%)
  pending 2/2, coverage 0.01%, no crashes yet

>>> fuzzer2 (0 days, 0 hrs) <<<

  cycle 1, lifetime speed 1 execs/sec, path 0/2 (0%)
  pending 2/2, coverage 0.01%, no crashes yet

Summary stats
=============

       Fuzzers alive : 2
      Total run time : 0 days, 0 hours
         Total execs : 0 million
    Cumulative speed : 2 execs/sec
       Pending paths : 4 faves, 4 total
  Pending per fuzzer : 2 faves, 2 total (on average)
       Crashes found : 0 locally unique

lazenca0x0@ubuntu:~/Documents/AFL/test$
```

Example

Example code

- .
  - ID, Password .
  - "Success" .
  - "Fail" .
- .
  - Stack Buffer Overflow .

test.c

```
#include <stdio.h>
#include <string.h>

int main(void){
        char login[16];
        char password[16];

        printf("Login : ");
        scanf("%s",login);
        printf("Password : ");
        scanf("%s",password);

        if(strcmp(login,"root") == 0){
                if(strcmp(password,"toor") == 0){
                        printf("Success.\n");
                        return 0;
                }
        }
        printf("Fail.\n");
        return 1;
}
```

Create to Test cases.

- **Test case .**
  - ID
  - Password
  - ID, Password
  - ID, Password

Create to test cases.

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ mkdir testcase
lazenca0x0@ubuntu:~/Documents/AFL/test$ cd testcase
lazenca0x0@ubuntu:~/Documents/AFL/test$ echo -e "a\toor" > test1.txt
lazenca0x0@ubuntu:~/Documents/AFL/test$ echo -e "root\na" > test2.txt
lazenca0x0@ubuntu:~/Documents/AFL/test$ echo -e "a\na" > test3.txt
lazenca0x0@ubuntu:~/Documents/AFL/test$ echo -e "root\toor" > test4.txt
lazenca0x0@ubuntu:~/Documents/AFL/test$
```

White-box testing

Build using afl-gcc.

- **AFL .**
  - .
  - Canary . (-fno-stack-protector)

afl-gcc -o test test.c

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-gcc -fno-stack-protector -o test test.c
afl-cc 2.49b by <lcamtuf@google.com>
test.c: In function 'main':
test.c:9:2: warning: ignoring return value of 'scanf', declared with attribute warn_unused_result [-Wunused-
result]
  scanf("%s",login);
  ^
test.c:11:2: warning: ignoring return value of 'scanf', declared with attribute warn_unused_result [-Wunused-
result]
  scanf("%s",password);
  ^
afl-as 2.49b by <lcamtuf@google.com>
[+] Instrumented 8 locations (64-bit, non-hardened mode, ratio 100%).
lazenca0x0@ubuntu:~/Documents/AFL/test$ ./test
Login : root
Password : toor
Success.
lazenca0x0@ubuntu:~/Documents/AFL/test$ ./test
Login : a
Password : a
Fail.
lazenca0x0@ubuntu:~/Documents/AFL/test$
```

Run afl-fuzz

- **AFL "uniq crashes" .**
  - 2 Uniq crashes .
- .
  - -i : Test case
  - -o :

Run AFL-fuzz

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ echo core > /proc/sys/kernel/core_pattern
lazenca0x0@ubuntu:~/Documents/AFL/test$ mkdir result
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-fuzz -i testcase/ -o result/ ./test
afl-fuzz 2.49b by <lcamtuf@google.com>
[+] You have 1 CPU core and 2 runnable tasks (utilization: 200%).
[*] Checking core_pattern...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning 'testcase/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:test1.txt'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 4, map size = 34, exec speed = 1428 us
[*] Attempting dry run with 'id:000001,orig:test2.txt'...
    len = 7, map size = 37, exec speed = 596 us
[*] Attempting dry run with 'id:000002,orig:test3.txt'...
    len = 14, map size = 38, exec speed = 740 us
[+] All test cases processed.

[+] Here are some useful stats:

    Test case count : 3 favored, 0 variable, 3 total
       Bitmap range : 34 to 38 bits (average: 36.33 bits)
        Exec timing : 596 to 1428 us (average: 921 us)

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

                        american fuzzy lop 2.49b (test)

 process timing  overall results
        run time : 0 days, 0 hrs, 0 min, 17 sec         cycles done : 16
   last new path : none yet (odd, check syntax!)        total paths : 3
 last uniq crash : 0 days, 0 hrs, 0 min, 11 sec        uniq crashes : 2
  last uniq hang : none seen yet                         uniq hangs : 0
 cycle progress  map coverage
  now processing : 1 (33.33%)            map density : 0.06% / 0.07%
 paths timed out : 0 (0.00%)          count coverage : 1.00 bits/tuple
 stage progress  findings in depth
  now trying : havoc                  favored paths : 3 (100.00%)
 stage execs : 136/256 (53.12%)        new edges on : 3 (100.00%)
 total execs : 29.8k                 total crashes : 242 (2 unique)
  exec speed : 1729/sec               total tmouts : 0 (0 unique)
 fuzzing strategy yields  path geometry
   bit flips : 0/176, 0/173, 0/167                     levels : 1
  byte flips : 0/22, 0/19, 0/13                      pending : 0
 arithmetics : 0/1228, 0/148, 0/0                   pend fav : 0
  known ints : 0/118, 0/532, 0/572                 own finds : 0
  dictionary : 0/0, 0/0, 0/24                       imported : n/a
       havoc : 2/13.6k, 0/12.9k                    stability : 100.00%
        trim : 14.29%/4, 0.00%
^C             [cpu:313%]

+++ Testing aborted by user +++
[+] We're done here. Have a nice day!
lazenca0x0@ubuntu:~/Documents/AFL/test$
```

Black-box testing

Install library files

- **Black box .**
  - **:** libini-config-dev, libtool-bin, automake, bison, libglib2.0-dev, qemu

Install library files

```
lazenca0x0@ubuntu:~/Documents/AFL/afl-2.49b$ apt-get install libini-config-dev libtool-bin automake bison
libglib2.0-dev qemu -y
lazenca0x0@ubuntu:~/Documents/AFL/afl-2.49b$ cd qemu_mode/
lazenca0x0@ubuntu:~/Documents/AFL/afl-2.49b/qemu_mode/$ ./build_qemu_support.sh
lazenca0x0@ubuntu:~/Documents/AFL/afl-2.49b/qemu_mode/$ cd ..
lazenca0x0@ubuntu:~/Documents/AFL/afl-2.49b$ sudo make install
```

Build using gcc

- **gcc .**

Build using gcc

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ gcc -fno-stack-protector -o test test.c
```

Run afl-fuzz

- **Black box test .**
  - Black box test -Q .
  - White box test 2 uniq crashes .

Run AFL-fuzz

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ afl-fuzz -Q -i testcase/ -o result/ ./test
afl-fuzz 2.49b by <lcamtuf@google.com>
[+] You have 1 CPU core and 3 runnable tasks (utilization: 300%).
[*] Checking core_pattern...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning 'testcase/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:test1.txt'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 4, map size = 33, exec speed = 1898 us
[*] Attempting dry run with 'id:000001,orig:test2.txt'...
    len = 6, map size = 33, exec speed = 1048 us
[!] WARNING: No new instrumentation output, test case may be useless.
[*] Attempting dry run with 'id:000002,orig:test3.txt'...
    len = 9, map size = 36, exec speed = 790 us
[*] Attempting dry run with 'id:000003,orig:test4.txt'...
    len = 6, map size = 33, exec speed = 806 us
[!] WARNING: No new instrumentation output, test case may be useless.
[+] All test cases processed.

[!] WARNING: Some test cases look useless. Consider using a smaller set.
[+] Here are some useful stats:

    Test case count : 2 favored, 0 variable, 4 total
       Bitmap range : 33 to 36 bits (average: 33.75 bits)
        Exec timing : 790 to 1898 us (average: 1135 us)

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

                        american fuzzy lop 2.49b (test)

 process timing  overall results
        run time : 0 days, 0 hrs, 0 min, 10 sec         cycles done : 5
   last new path : none yet (odd, check syntax!)        total paths : 4
 last uniq crash : 0 days, 0 hrs, 0 min, 2 sec         uniq crashes : 2
  last uniq hang : none seen yet                         uniq hangs : 0
 cycle progress  map coverage
  now processing : 1* (25.00%)           map density : 0.05% / 0.06%
 paths timed out : 0 (0.00%)          count coverage : 1.00 bits/tuple
 stage progress  findings in depth
  now trying : splice 7               favored paths : 2 (50.00%)
 stage execs : 30/32 (93.75%)          new edges on : 2 (50.00%)
 total execs : 16.2k                  total crashes : 1204 (2 unique)
  exec speed : 1533/sec                total tmouts : 0 (0 unique)
 fuzzing strategy yields  path geometry
   bit flips : 0/128, 0/124, 0/116                     levels : 1
  byte flips : 0/16, 0/12, 0/4                       pending : 0
 arithmetics : 0/890, 0/176, 0/0                    pend fav : 0
  known ints : 0/80, 0/336, 0/176                  own finds : 0
  dictionary : 0/0, 0/0, 0/2                        imported : n/a
       havoc : 1/7936, 1/6184                      stability : 100.00%
        trim : 42.86%/4, 0.00%
             [cpu:303%]

+++ Testing aborted by user +++
[+] We're done here. Have a nice day!

lazenca0x0@ubuntu:~/Documents/AFL/test$
```

Check for the crash.

- uniq **crashes result .**
  - crash .

Check for the crash.

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ ls -al result/crashes/
total 20
drwx------ 2 lazenca0x0 lazenca0x0 4096 Aug  9 01:26 .
drwxrwxr-x 5 lazenca0x0 lazenca0x0 4096 Aug  9 01:26 ..
-rw------- 1 lazenca0x0 lazenca0x0   68 Aug  9 01:26 id:000000,sig:11,src:000000,op:havoc,rep:128
-rw------- 1 lazenca0x0 lazenca0x0   86 Aug  9 01:26 id:000001,sig:11,src:000002+000003,op:splice,rep:128
-rw------- 1 lazenca0x0 lazenca0x0  604 Aug  9 01:26 README.txt
lazenca0x0@ubuntu:~/Documents/AFL/test$ ./test < result/crashes/id\:000000\,sig\:11\,src\:000000\,op\:havoc\,
rep\:128
Login : Password : Fail.
Segmentation fault
lazenca0x0@ubuntu:~/Documents/AFL/test$ ./test < result/crashes/id\:000001\,sig\:11\,src\:000002+000003\,op\:
splice\,rep\:128
Login : Password : Fail.
Segmentation fault
lazenca0x0@ubuntu:~/Documents/AFL/test$
```

- **crash .**
  - .

Crash file

```
lazenca0x0@ubuntu:~/Documents/AFL/test$ hexdump result/crashes/id\:000000\,sig\:11\,src\:000000\,op\:havoc\,
rep\:128
0000000 81b9 ad13 0000 76e1 04ff 007f eee7 ffff
0000010 64ff 0000 798a 9379 7980 7979 7966 e100
0000020 ff76 7fc0 e700 ffee ffff ffff 7f04 e700
0000030 ffee ffff 0064 6900 7979 7993 7979 0079
0000040 0100 ff00
0000044
lazenca0x0@ubuntu:~/Documents/AFL/test$ hexdump result/crashes/id\:000001\,sig\:11\,src\:000002+000003\,op\:
splice\,rep\:128
0000000 6f72 746f 0000 0004 5774 aaaa aaaa aaaa
0000010 aa97 aaaa 0000 8000 5774 aaaa 97a4 aaaa
0000020 00aa 0000 7480 aa57 9faa 72aa 6f6f aa74
0000030 aaaa 97a4 aaaa 16aa aaaa 619c aa57 aaaa
0000040 aaaa 97aa aaaa 00aa 0000 aa80 6f6f aaaa
0000050 72aa 6f6f 6f74
0000056
```

Related information

- http://lcamtuf.coredump.cx/afl/README.txt
- http://lcamtuf.coredump.cx/afl/QuickStartGuide.txt
- http://lcamtuf.coredump.cx/afl/technical_details.txt
- https://lcamtuf.blogspot.jp/2014/10/fuzzing-binaries-without-execve.html
- https://lcamtuf.blogspot.jp/2016/02/say-hello-to-afl-analyze.html
- https://raw.githubusercontent.com/mirrorer/afl/master/docs/parallel_fuzzing.txt

Unknown macro: 'html'