Fuzzing Fuzzgoat with AFL++

AFL

AFL - American Fuzzy Lop
Developed by - Michal Zalewski
For installing afl++
 # git clone https://github.com/AFLplusplus/AFLplusplus.git
 # make
 # sudo make install

What it 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.

Download the source of fuzzgoat with:

 # git clone https://github.com/fuzzstati0n/fuzzgoat.git

```
root@kali:~# git clone https://github.com/fuzzstati0n/fuzzgoat.git
Cloning into 'fuzzgoat'...
remote: Enumerating objects: 127, done.
remote: Total 127 (delta 0), reused 0 (delta 0), pack-reused 127
Receiving objects: 100% (127/127), 26.40 KiB | 259.00 KiB/s, done.
Resolving deltas: 100% (61/61), done.
```

Building Fuzzgoat:

For output create a output folder.
#mkdir out
Go to the path of fuzzgoat and the build it with make.

```
root@kali:~/Desktop/fuzzing/fuzzgoat# make
afl-gcc -o fuzzgoat -I. main.c fuzzgoat.c -lm
afl-cc++2.66c by Michal Zalewski
[!] NOTE: afl-gcc is deprecated, llvm_mode is much faster and has more options
afl-as++2.66c by Michal Zalewski
[+] Instrumented 75 locations (64-bit, non-hardened mode, ratio 100%).
afl-as++2.66c by Michal Zalewski
[+] Instrumented 372 locations (64-bit, non-hardened mode, ratio 100%).
afl-gcc -fsanitize=address -o fuzzgoat_ASAN -I. main.c fuzzgoat.c -lm
afl-cc++2.66c by Michal Zalewski
[!] NOTE: afl-gcc is deprecated, llvm_mode is much faster and has more options
afl-as++2.66c by Michal Zalewski
[+] Instrumented 45 locations (64-bit, non-hardened, ASAN mode, ratio 33%).
afl-as++2.66c by Michal Zalewski
[+] Instrumented 318 locations (64-bit, non-hardened, ASAN mode, ratio 33%).
root@kali:~/Desktop/fuzzing/fuzzgoat# ls
afl-command-line  fuzzgoat_ASAN  fuzzgoat.h        in           LICENSE  Makefile  README.md  ut
fuzzgoat          fuzzgoat.c     fuzzgoatNoVulns.c  input-files  main.c   out       seed
```

{in this case fuzzgoat is located at Desktop/fuzzing/fuzzgoat}

Get Set Fuzz!!

 # afl-fuzz -i in -o out ./fuzzgoat @@

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

```
root@kali:~/Desktop/fuzzing/fuzzgoat# afl-fuzz -i in -o out ./fuzzgoat @@
afl-fuzz++2.66c based on afl by Michal Zalewski and a big online community
[+] afl++ is maintained by Marc "van Hauser" Heuse, Heiko "hexcoder" Eißfeldt, Andrea Fioraldi and Dominik Maie
r
[+] afl++ is open source, get it at https://github.com/AFLplusplus/AFLplusplus
[+] Power schedules from github.com/mboehme/aflfast
[+] Python Mutator and llvm_mode instrument file list from github.com/choller/afl
[+] MOpt Mutator from github.com/puppet-meteor/MOpt-AFL
[*] Getting to work...
[+] Using exploration-based constant power schedule (EXPLORE, default)
[+] You have 4 CPU cores and 2 runnable tasks (utilization: 50%).
[+] Try parallel jobs - see /usr/local/share/doc/afl/parallel_fuzzing.md.
[*] Checking CPU core loadout...
[+] Found a free CPU core, try 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,time:0,orig:seed'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
[+] All test cases processed.
```

{The basic tests and checks before fuzzer start.}

Fuzzing and analysing the crashes:

```
            american fuzzy lop ++2.66c (fuzzgoat) [explore] {0}
— process timing —                                    — overall results —
        run time : 0 days, 0 hrs, 3 min, 6 sec        cycles done : 3
   last new path : 0 days, 0 hrs, 0 min, 0 sec        total paths : 321
 last uniq crash : 0 days, 0 hrs, 0 min, 9 sec       uniq crashes : 27
  last uniq hang : none seen yet                       uniq hangs : 0
— cycle progress —                                  — map coverage —
  now processing : 162*0 (50.5%)                      map density : 0.17% / 0.74%
 paths timed out : 0 (0.00%)                       count coverage : 2.62 bits/tuple
— stage progress —                                  — findings in depth —
  now trying : havoc                               favored paths : 87 (27.10%)
 stage execs : 5723/6144 (93.15%)                   new edges on : 121 (37.69%)
 total execs : 898k                                total crashes : 1962 (27 unique)
  exec speed : 4880/sec                             total tmouts : 0 (0 unique)
— fuzzing strategy yields —                                     — path geometry —
   bit flips : 19/10.1k, 10/9972, 7/9708                           levels : 9
  byte flips : 0/1263, 0/1131, 1/892                              pending : 166
 arithmetics : 44/70.2k, 0/8307, 0/412                           pend fav : 0
  known ints : 3/6893, 0/30.9k, 1/39.1k                         own finds : 320
  dictionary : 0/0, 0/0, 0/62                                    imported : n/a
havoc/splice : 260/700k, 0/0                                    stability : 100.00%
   py/custom : 0/0, 0/0
        trim : 29.86%/379, 19.72%                                   [cpu000: 75%]
                                                                ^Z
```

{3 cycles,total path,27 unique crashes were found}

Data under out{output} directory:

```
root@kali:~/Desktop/fuzzing/fuzzgoat# cd out
root@kali:~/Desktop/fuzzing/fuzzgoat/out# ls
cmdline  crashes  fuzz_bitmap  fuzzer_stats  hangs  plot_data  queue
```

Data under crashes:

```
root@kali:~/Desktop/fuzzing/fuzzgoat/out# cd crashes
root@kali:~/Desktop/fuzzing/fuzzgoat/out/crashes# ls
id:000000,sig:11,src:000000,time:147,op:arith8,pos:5,val:-5
id:000001,sig:06,src:000000,time:317,op:havoc,rep:8
id:000002,sig:11,src:000000,time:336,op:havoc,rep:2
id:000003,sig:06,src:000000,time:610,op:havoc,rep:4
id:000004,sig:11,src:000000,time:1407,op:havoc,rep:2
id:000005,sig:06,src:000000,time:4863,op:havoc,rep:8
id:000006,sig:11,src:000000,time:5596,op:havoc,rep:2
id:000007,sig:06,src:000003,time:8643,op:arith8,pos:3,val:+35
id:000008,sig:06,src:000003,time:9146,op:havoc,rep:2
id:000009,sig:06,src:000003,time:10919,op:havoc,rep:8
id:000010,sig:06,src:000003,time:10970,op:havoc,rep:2
id:000011,sig:06,src:000067,time:33695,op:havoc,rep:4
id:000012,sig:06,src:000153,time:53685,op:arith8,pos:3,val:-13
id:000013,sig:06,src:000153,time:53887,op:havoc,rep:2
id:000014,sig:06,src:000153,time:54964,op:havoc,rep:8
id:000015,sig:06,src:000153,time:56069,op:havoc,rep:8
id:000016,sig:11,src:000202,time:66966,op:havoc,rep:2
id:000017,sig:11,src:000217,time:86099,op:havoc,rep:2
id:000018,sig:06,src:000229,time:90599,op:havoc,rep:8
id:000019,sig:11,src:000230,time:97123,op:havoc,rep:4
id:000020,sig:11,src:000009,time:146977,op:flip1,pos:5
id:000021,sig:11,src:000009,time:147066,op:arith8,pos:5,val:+5
id:000022,sig:11,src:000009,time:147074,op:arith8,pos:5,val:-26
id:000023,sig:11,src:000009,time:147075,op:arith8,pos:5,val:-29
id:000024,sig:11,src:000009,time:147076,op:arith8,pos:5,val:-30
id:000025,sig:11,src:000116,time:166949,op:arith8,pos:37,val:-5
id:000026,sig:11,src:000071,time:177357,op:havoc,rep:16
README.txt
```

`.triage_crashes.sh`:-

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

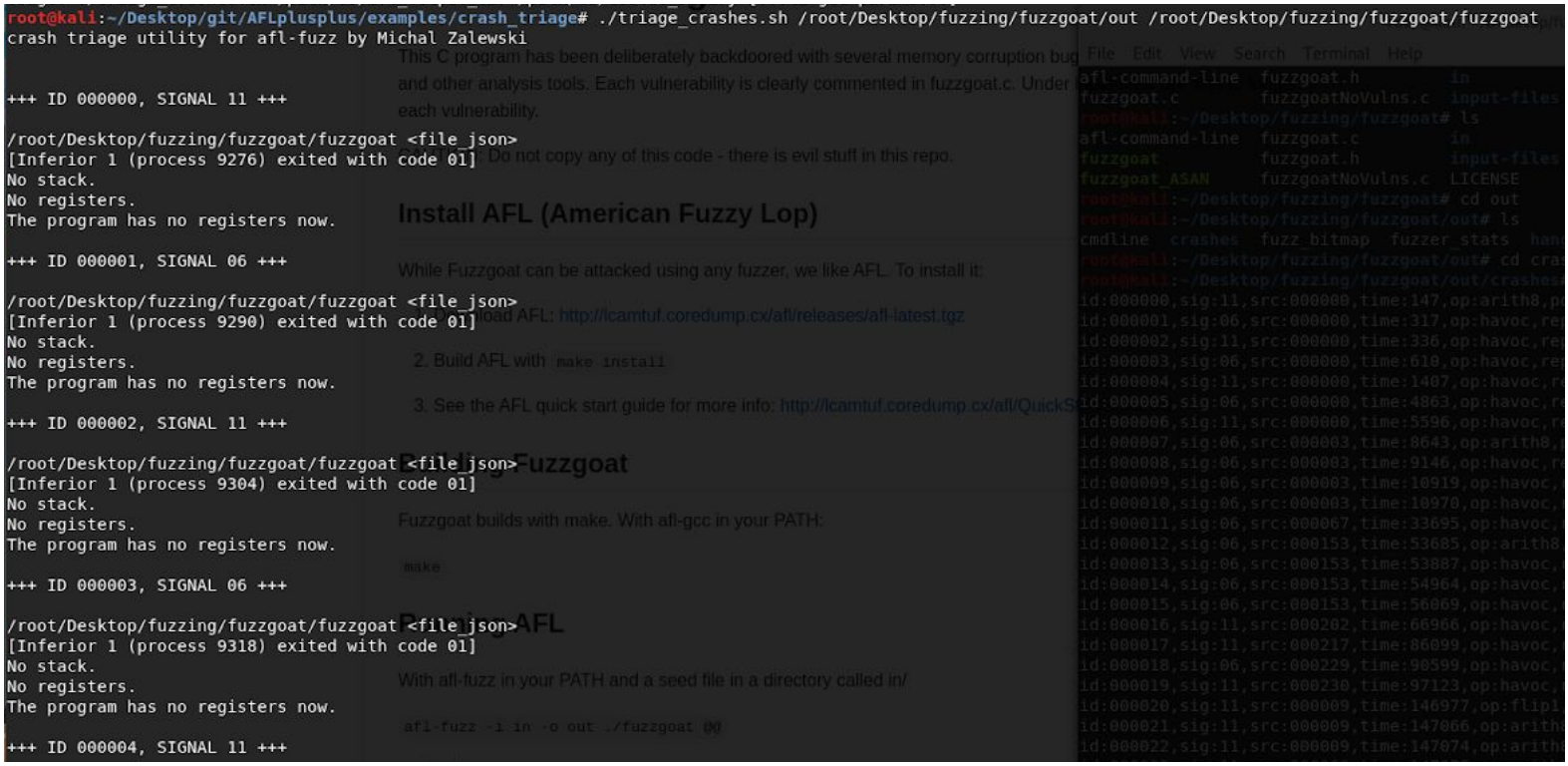
For this goto to the location where afl in installed

Then goto to experimental/crash_triage/

```
root@kali:~/Desktop/git/AFLplusplus/examples/crash_triage# ls
triage_crashes.sh
root@kali:~/Desktop/git/AFLplusplus/examples/crash_triage# ./triage_crashes.sh
crash triage utility for afl-fuzz by Michal Zalewski

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

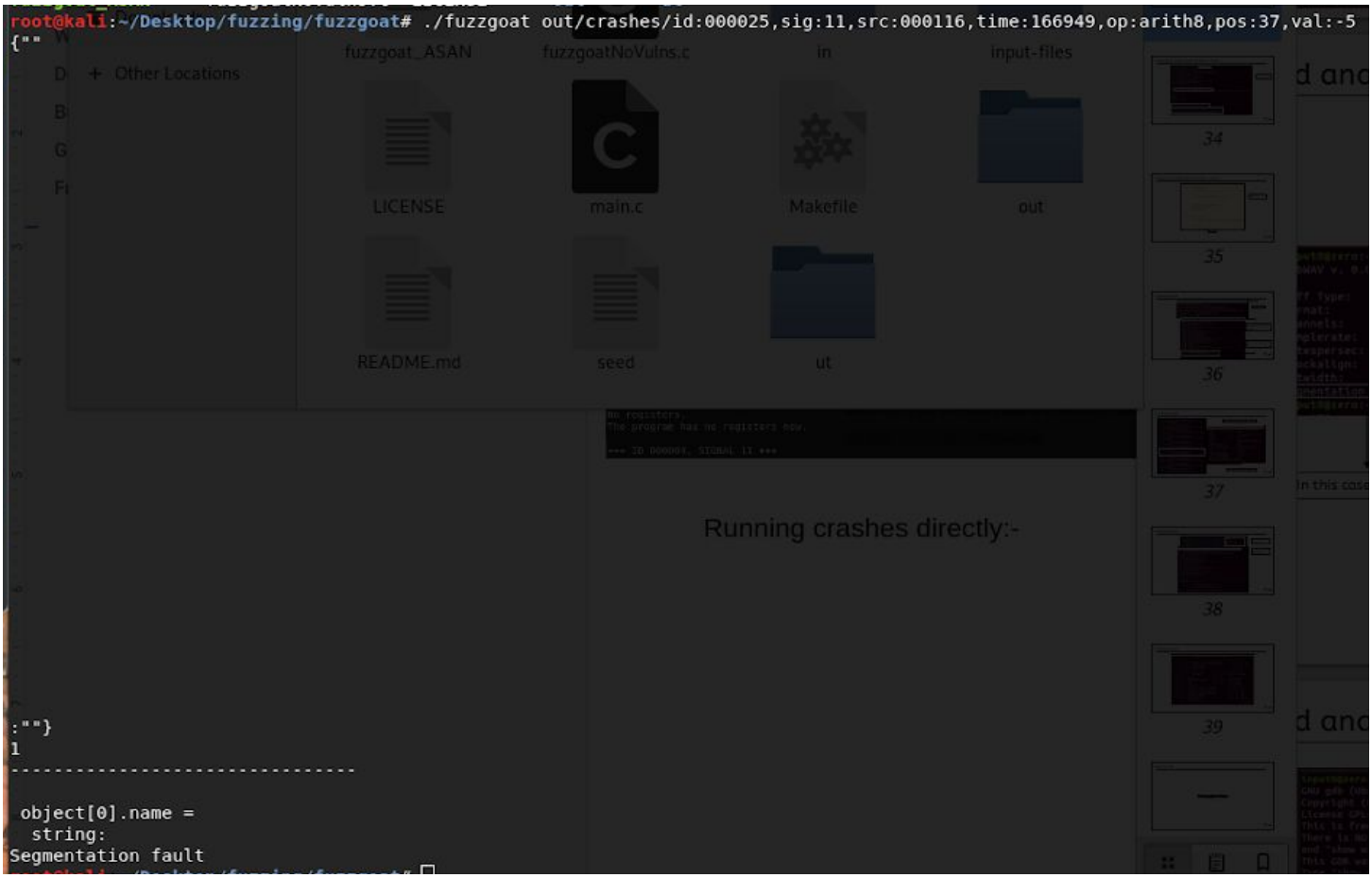
 # ./triage_crashes.sh /root/Desktop/fuzzing/fuzzgoat/out /root/Desktop/fuzzing/fuzzgoat/fuzzgoat



Running crashes directly:-

#./fuzzgoat/out/crashes/id:000025,sig:11,src:000116,time:166949,op:arith8, pos:37,val:-5

{id can be anything depending on which crash you want to analyse}



Analysing crash via GDB:-

 # gdb ./fuzzgoat

(gdb) run out/crashes/id:000025,sig:11,src:000116,time:166949,op:arith8,pos:37,val:5

```
(gdb) run out/crashes/id:000025,sig:11,src:000116,time:166949,op:arith8,pos:37,val:-5
```

This is the cause for the segmentation fault.

```
Starting program: /root/Desktop/fuzzing/fuzzgoat/fuzzgoat out/crashes/id:000025,sig:11,src:000116,time:166949,op:arith8,pos:37,val:-
{""
:""}
1
-----------------------------

 object[0].name =
  string:

Program received signal SIGSEGV, Segmentation fault.
0x0000555555557b40 in json_value_free_ex (settings=settings@entry=0x7fffffffdf30, value=0x21) at fuzzgoat.c:224
224           switch (value->type)
(gdb)
```