OT Lab 3 - Fuzzing

Student: *Ahmed Elkashef*

1. Playing with AFL:

Use AFL on any old source code and generate crashes. Verify these crashes. Explain what is happening from the crashes generated.

I will use AFL on fuzzgoat, which is an intentionally vulnerable application written to demonstrate fuzzing.

I will clone and cd into the directory of fuzzgoat:

```
git clone https://github.com/fuzzstati0n/fuzzgoat
cd fuzzgoat/
```

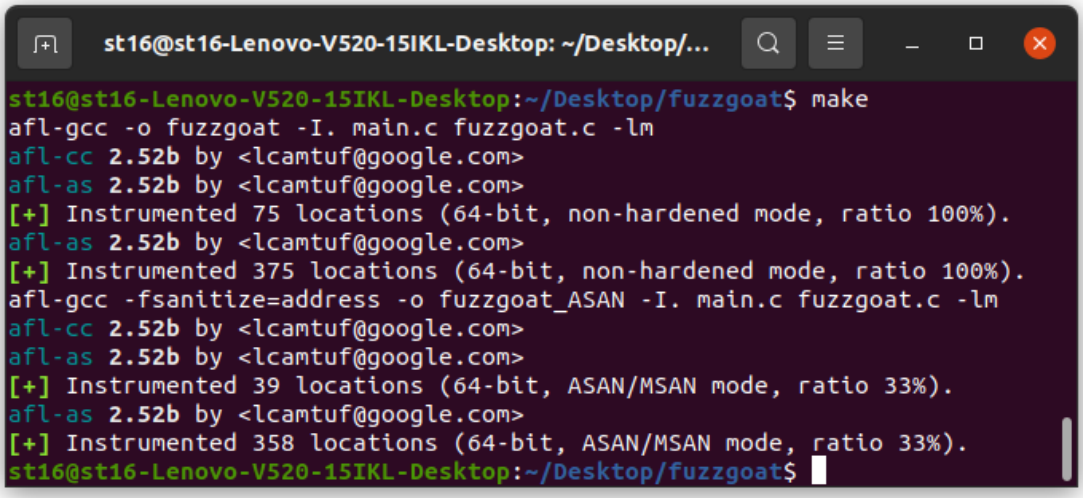
Then compile the binary with one of AFL's compilers:

```
export CC=afl-clang-fast
```

Noting that we can use any of the following:

- afl-clang-fast
- afl-clang-fast++
- afl-gcc
- afl-g++

Then we compile fuzzgoat into a binary:



The next step is the test corpus, which is basically a "seed input file", they are the starting point for AFL to mutate and generate new test files as it sees fit to discover new code paths.

I will start by making a directory, and piping some random, garbage value to the directory:

```
mkdir afl_in
cp /bin/ps afl_in/
```

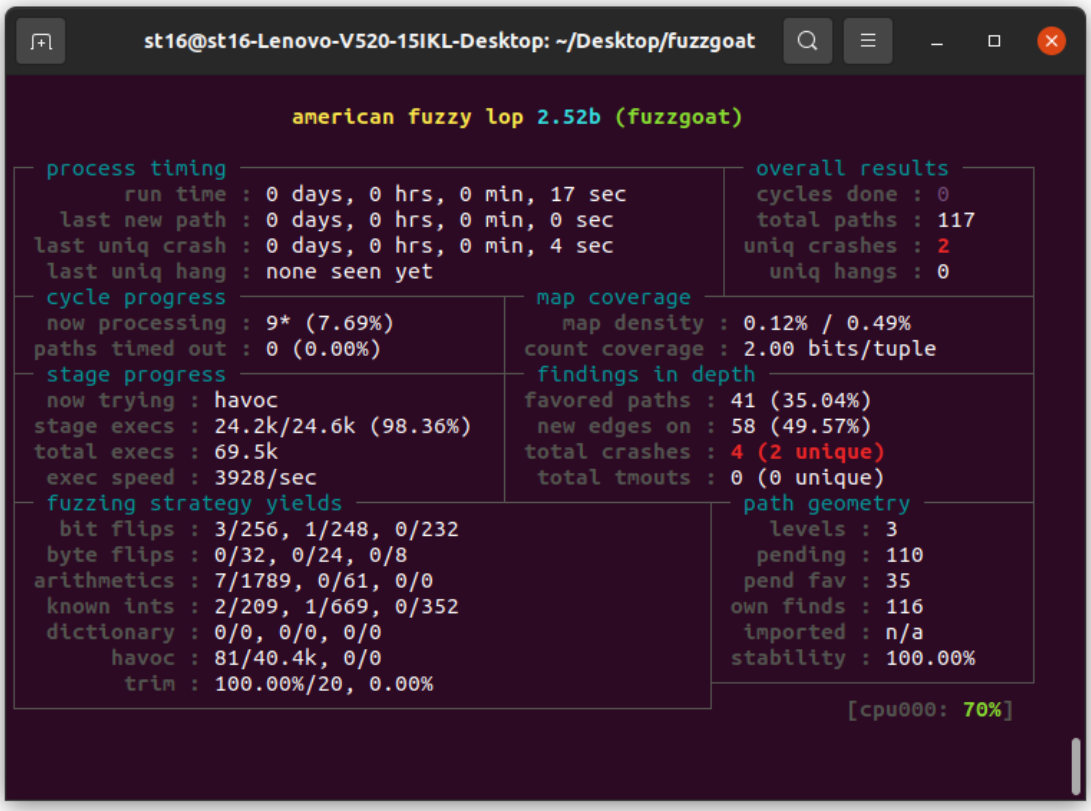
The last thing is to specify where are the files that have resulted in a crash or a hang will be stored, let's call it afl_out:

```
mkdir afl_out
```

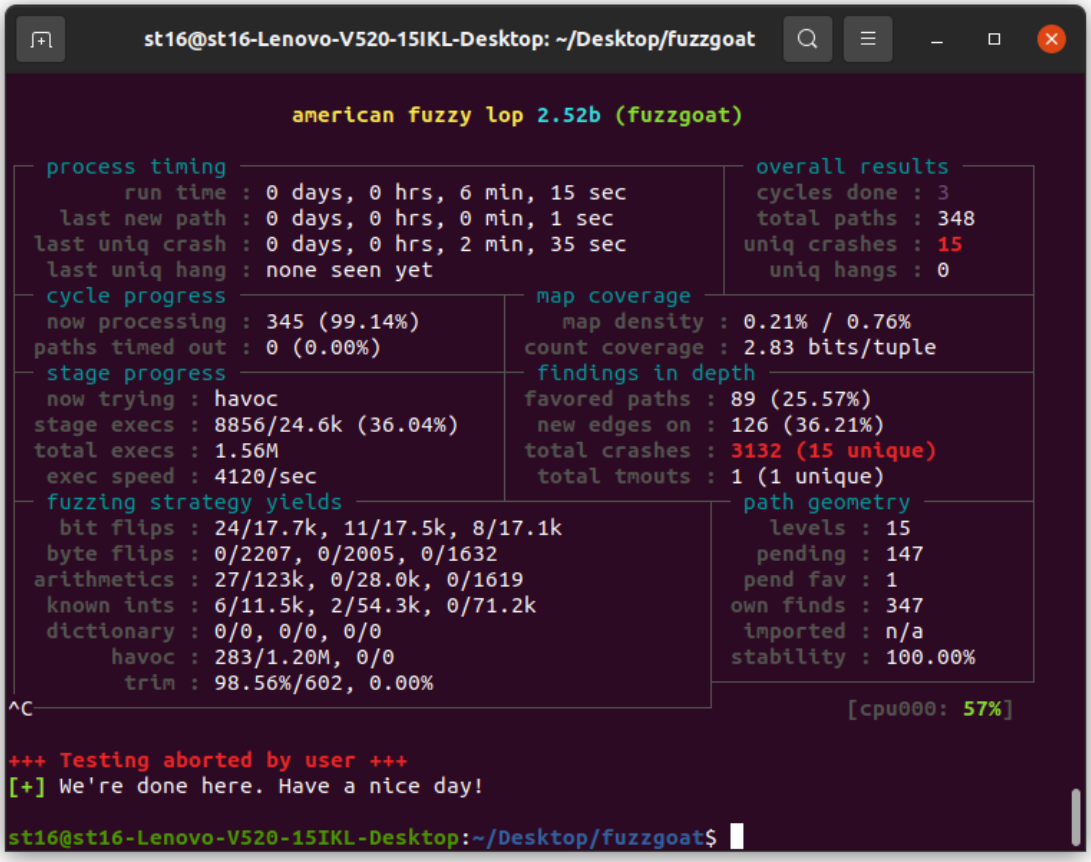
The last thing is to finally fuzz the application:

```
afl-fuzz -i afl_in -o afl_out -- ./fuzzgoat @@
```

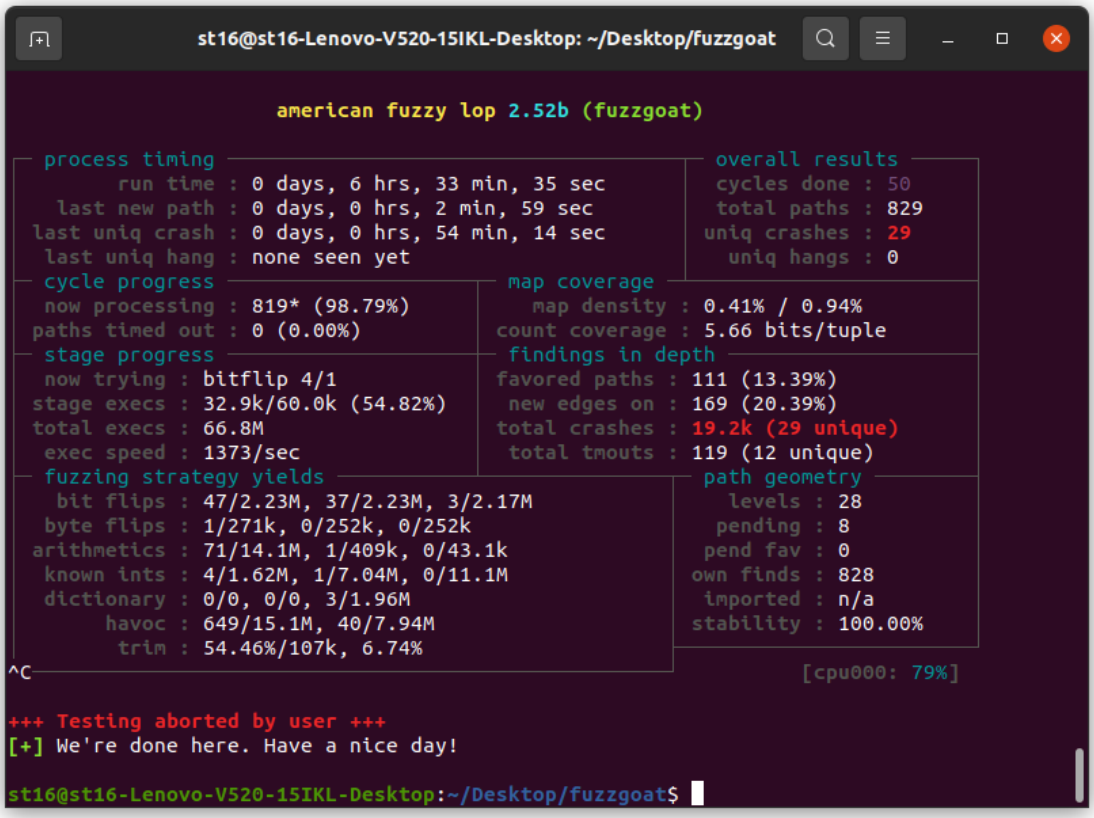
1) -i specifies the input seed files
2) -o specifies the output files directory
3) -- separates the arguments from the target
4) ./fuzzgoat is the target
5) @@ is the position where AFL is supposed to insert the test file in the target application's command structure



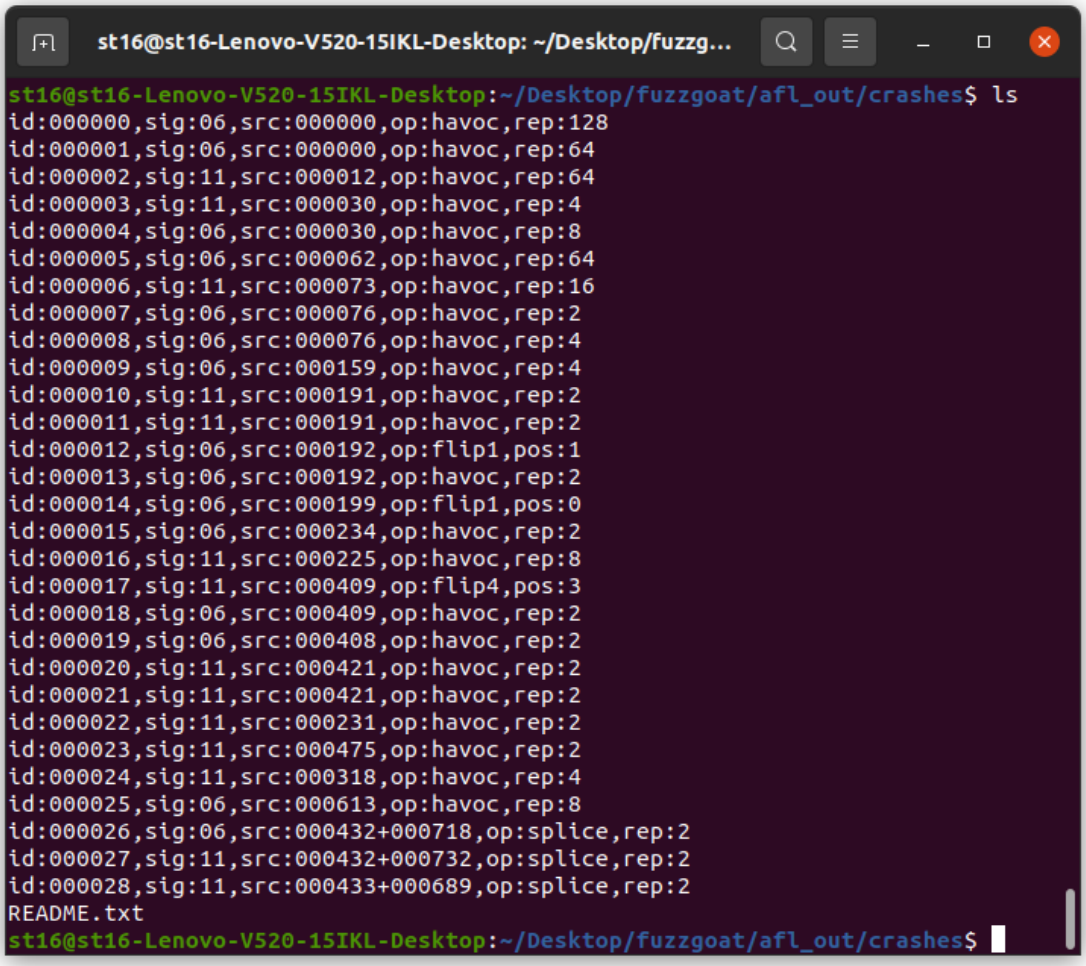
I stop the fuzzing after I have generated some crashes:



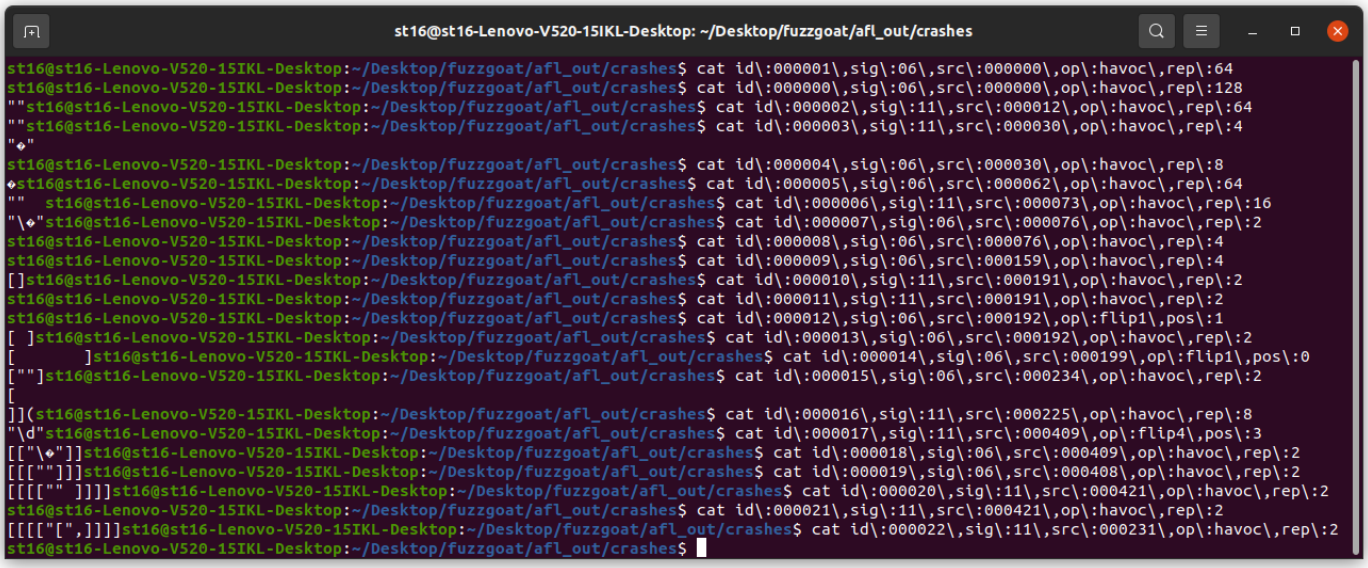
It is recommended to stop the fuzzer after the cycles done are 50:



Then I navigate to the afl_out directory -> crashes and check its contents:

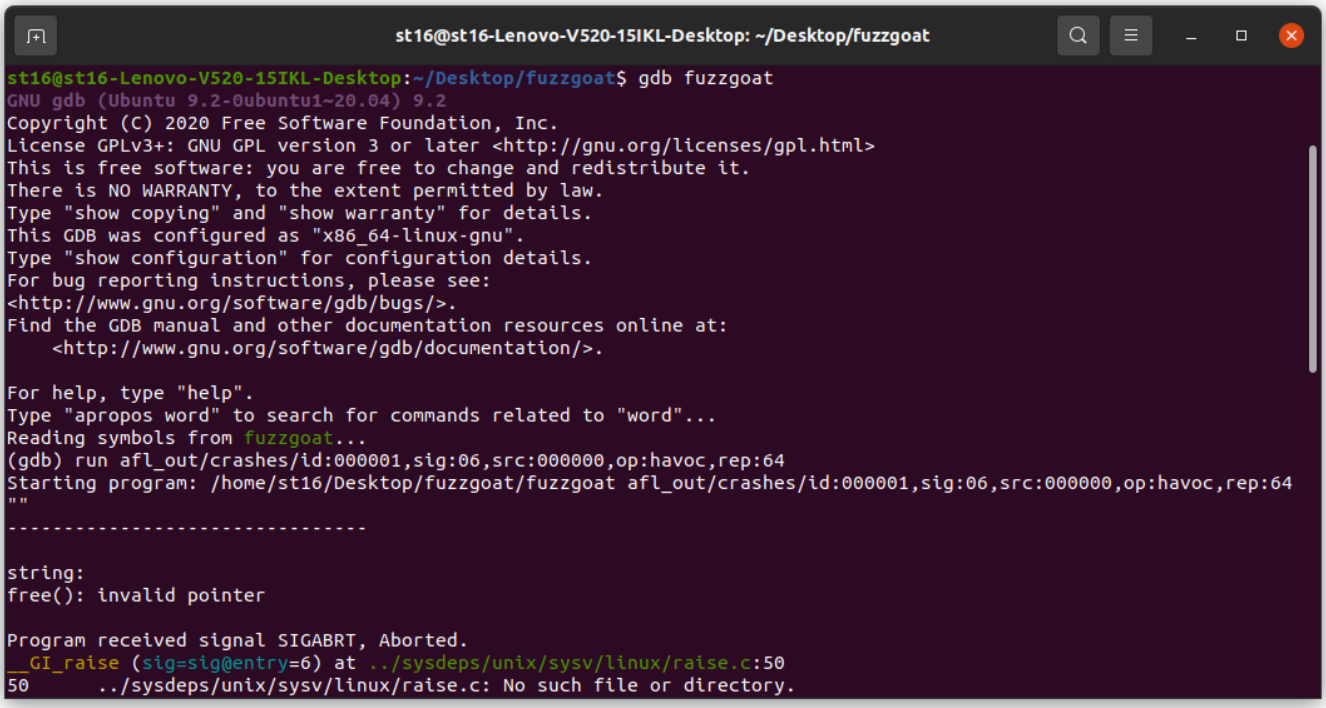


To have a look at these crashes and see what triggered it:



The list of characters are emphasized when I cat these files.

Now, I will run the program with gdb and pass in the crash file and see for more details:




```
string:
free(): invalid pointer

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50     ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
```

Another crash we can backtrace with bt on gdb:

```
(gdb) run afl_out/crashes/id:000009,sig:06,src:000159,op:havoc,rep:4
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/st16/Desktop/fuzzgoat/fuzzgoat afl_out/crashes/id:000009,sig:06,src:000159,op:havoc,rep:4
[]
------------------------------

free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50      ../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:50
#1  0x00007ffff7c8e859 in __GI_abort () at abort.c:79
#2  0x00007ffff7cf93ee in __libc_message (action=action@entry=do_abort, fmt=fmt@entry=0x7ffff7e23285 "%s\n")
    at ../sysdeps/posix/libc_fatal.c:155
#3  0x00007ffff7d0147c in malloc_printerr (
    str=str@entry=0x7ffff7e255d0 "free(): double free detected in tcache 2") at malloc.c:5347
#4  0x00007ffff7d030ed in _int_free (av=0x7ffff7e54b80 <main_arena>, p=0x5555555628a0, have_lock=0)
    at malloc.c:4201
#5  0x0000555555557c9a in json_value_free_ex (settings=settings@entry=0x7fffffffd7d8, value=0x0) at fuzzgoat.c:311
#6  0x0000555555558c8c in json_value_free_ex (value=<optimized out>, settings=0x7fffffffd7d8) at fuzzgoat.c:222
#7  json_parse_ex (settings=settings@entry=0x7fffffffd8f0, json=<optimized out>, length=<optimized out>,
    error_buf=error_buf@entry=0x0) at fuzzgoat.c:1065
#8  0x000055555555e295 in json_parse (json=<optimized out>, length=<optimized out>) at fuzzgoat.c:1073
#9  0x000055555555567b in main (argc=<optimized out>, argv=<optimized out>) at main.c:156
(gdb)
```

```
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50     ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
```

Looking at the backtrace, it looks like a free() function that was called twice on the same pointer, or calling a free on a null pointer.
Looking at the code in fuzzgoat.c, the origin of the problem is traced to here:

```
/*****************************************************************************
  WARNING: Fuzzgoat Vulnerability

  The code below creates and dereferences a NULL pointer if the string
  is of length one.

  Diff       - Check for one byte string - create and dereference a NULL pointer
  Payload    - A JSON string of length one : "A"
  Input File - oneByteString
  Triggers   - NULL pointer dereference
*****************************************************************************/

            if (value->u.string.length == 1) {
              char *null_pointer = NULL;
              printf ("%d", *null_pointer);
            }
/****** END vulnerable code ************************************************/

            settings->mem_free (value->u.string.ptr, settings->user_data);
            break;

         default:
            break;
      };
```

- **References:**

1. https://forums.gentoo.org/viewtopic-t-1098826-start-0.html
2. https://stackoverflow.com/questions/29382571/seg-fault-signal-received-sigabort
3. https://github.com/fuzzstati0n/fuzzgoat
4. https://github.com/google/AFL
5. https://medium.com/@ayushpriya10/fuzzing-applications-with-american-fuzzy-lop-afl-54facc65d102
6. https://www.reddit.com/r/programminghelp/comments/juqrrp/c_free_double_free_detected_in_tcache_2/
7. https://github.com/fuzzstati0n/fuzzgoat/issues/1
8. https://www.delftstack.com/howto/c/c-free-invalid-pointer-error/
9.