

Assignment 5

IDATT2503 - Security in programming and cryptography
Fall 2023

Francesca Grimaldi

1 Fuzzing

Following the instructions of the [fuzzing-example](#), I executed the following steps:

1. Installed `cmake` and `clang` from terminal
2. Created a `build` directory inside that of my C project
3. Moved into the new build directory
4. Configured the build system for the project: set the C compiler to be Clang and ran the CMake build system
5. Compiled the source code and created executable files based on the `CMakeLists.txt` ones
6. Ran the fuzzer test with a time limit of 60 seconds
7. Ran the utility tests with assertions

```
fragrimi@DESKTOP-TPG3QA:~$ mkdir c/build
fragrimi@DESKTOP-TPG3QA:~$ cd c/build
fragrimi@DESKTOP-TPG3QA:~/c/build$ CC=clang cmake ..
-- The C compiler identification is Clang 10.0.0
-- Check for working C compiler: /usr/bin/clang
-- Check for working C compiler: /usr/bin/clang -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/fragrimi/c/build
fragrimi@DESKTOP-TPG3QA:~/c/build$ make
Scanning dependencies of target utility
[ 11%] Building C object CMakeFiles/utility.dir/utility.c.o
[ 22%] Linking C static library libutility.a
[ 22%] Built target utility
Scanning dependencies of target c
[ 33%] Building C object CMakeFiles/c.dir/main.c.o
```

Figure 1: Initial setup

The first fuzzing showed that there were memory leaks in the program.

```
fragrimi@DESKTOP-TPG3QA:~/c$ ./build/tests/utility_fuzzer_test -max_total_time=60
INFO: Seed: 1051916668
INFO: Loaded 1 modules (1 inline 8-bit counters): 1 [0x5a3eb0, 0x5a3eb1],
INFO: Loaded 1 PC tables (1 PCs): 1 [0x567ef8, 0x567f00],
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2 INITED cov: 1 ft: 1 corp: 1/1b exec/s: 0 rss: 26Mb

=====
==1881==ERROR: LeakSanitizer: detected memory leaks
Direct leak of 2 byte(s) in 1 object(s) allocated from:
#0 0x51e1dd in malloc (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x51e1dd)
#1 0x54daea in replace_char (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x54daea)
#2 0x54d999 in LLVMFuzzerTestOneInput (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x54d999)
#3 0x458671 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x458671)
)
#4 0x457db5 in fuzzer::Fuzzer::RunOne(unsigned char const*, unsigned long, bool, fuzzer::InputInfo*, bool*) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x457db5)
#5 0x45a591 in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::__Fuzzer::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> > &) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x45a591)
#6 0x45aa39 in fuzzer::Fuzzer::Loop(std::__Fuzzer::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> > &) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x45aa39)
#7 0x44970e in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x44970e)
#8 0x472552 in main (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x472552)
#9 0x7f340d78082 in __libc_start_main /build/glibc-SzIz7B/glibc-2.31/csu/../csu/libc-start.c:308:16
Direct leak of 2 byte(s) in 1 object(s) allocated from:
#0 0x51e1dd in malloc (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x51e1dd)
#1 0x54daea in replace_char (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x54daea)
#2 0x54d999 in LLVMFuzzerTestOneInput (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x54d999)
#3 0x458671 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) (/home/fragrimi/c/build/tests/utility_fuzzer_test+0x458671)
)
```

Figure 2: Fuzzing showing memory leaks (a)

```

)
#4 0x457db5 in fuzzer::Fuzzer::RunOne(unsigned char const*, unsigned long, bool, fuzzer::InputInfo*, bool*) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x457db5)
#5 0x45a057 in fuzzer::Fuzzer::MutateAndTestOne() (/home/fragrim/c/build/tests/utility_fuzzer_test+0x45a057)
#6 0x45ad55 in fuzzer::Fuzzer::Loop(std::__Fuzzer::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >6) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x45ad55)
#7 0x44970e in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x44970e)
#8 0x472552 in main (/home/fragrim/c/build/tests/utility_fuzzer_test+0x472552)
#9 0x7f340df78082 in __libc_start_main /build/glibc-SzIz7B/glibc-2.31/csu/../csu/libc-start.c:308:16

Direct leak of 1 byte(s) in 1 object(s) allocated from:
#0 0x51e1dd in malloc (/home/fragrim/c/build/tests/utility_fuzzer_test+0x51e1dd)
#1 0x54daea in replace_char (/home/fragrim/c/build/tests/utility_fuzzer_test+0x54daea)
#2 0x54d999 in LLVMFuzzerTestOneInput (/home/fragrim/c/build/tests/utility_fuzzer_test+0x54d999)
#3 0x458671 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x458671)
)
#4 0x45a3aa in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::__Fuzzer::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >6) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x45a3aa)
#5 0x45a3a9 in fuzzer::Fuzzer::Loop(std::__Fuzzer::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >6) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x45a3a9)
#6 0x44970e in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) (/home/fragrim/c/build/tests/utility_fuzzer_test+0x44970e)
#7 0x472552 in main (/home/fragrim/c/build/tests/utility_fuzzer_test+0x472552)
#8 0x7f340df78082 in __libc_start_main /build/glibc-SzIz7B/glibc-2.31/csu/../csu/libc-start.c:308:16

SUMMARY: AddressSanitizer: 5 byte(s) leaked in 3 allocation(s).
INFO: to ignore leaks on libFuzzer side use -detect_leaks=0.

MS: 1 ShuffleBytes-; base unit: adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
0xa,
\x0a
artifact_prefix='./; Test unit written to ./leak-adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
Base64: Cg==
fragrimi@DESKTOP-TTPG3QA:~/c$

```

Figure 3: Fuzzing showing memory leaks (b)

After the bug fixes, the results are the following:

```

fragrimi@DESKTOP-TTPG3QA:~/c/build$ ./tests/utility_fuzzer_test -max_total_time=60
INFO: Seed: 2471828778
INFO: Loaded 1 modules (3 inline 8-bit counters): 3 [0x5a3eb0, 0x5a3eb3],
INFO: Loaded 1 PC tables (3 PCs): 3 [0x567ef8, 0x567f28],
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2 INITED cov: 2 ft: 2 corp: 1/1b exec/s: 0 rss: 260Mb
#1848576 pulse cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 262144 rss: 811Mb
#2897152 pulse cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 161319 rss: 811Mb
#4194384 pulse cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 119837 rss: 811Mb
#6332827 DONE cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 103816 rss: 811Mb
Done 6332827 runs in 61 second(s)
fragrimi@DESKTOP-TTPG3QA:~/c/build$

```

Figure 4: Fuzzing after fixes

```

fragrimi@DESKTOP-TTPG3QA:~/c/build$ ./tests/utility_test
fragrimi@DESKTOP-TTPG3QA:~/c/build$

```

Figure 5: Running tests does not produce errors

2 CI solution

To set up a Continuous Integration solution for fuzzing with address sanitizer, I used the GitHub Actions platform.

For this, I created a repository with the project and a `.github/workflows` folder, in which I added the YAML file `ci-fuzzing.yml`. In this I defined the workflow, including the instructions necessary for the setup and those for the fuzzing and testing.

After committing the above-mentioned file to the repository, it gives the following output:

```

build
Started 18s ago
Search logs
Initial Setup 3s
36 [100%] Linking C executable utility_fuzzer_test
37 [100%] Built target utility_fuzzer_test
Fuzzing 6s
1 ▶ Run cd build
5 INFO: Running with entropic power schedule (0xFF, 100).
6 INFO: Seed: 2128748680
7 INFO: Loaded 1 modules (3 inline 8-bit counters): 3 [0x55e9530b2ef0, 0x55e9530b2ef3],
8 INFO: Loaded 1 PC tables (3 PCs): 3 [0x55e9530b2ef8, 0x55e9530b2f28],
9 INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
10 INFO: A corpus is not provided, starting from an empty corpus
11 #2 INITED cov: 2 ft: 2 corp: 1/1b exec/s: 0 rss: 320Mb
12 #1848576 pulse cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 524288 rss: 250Mb
13 #2897152 pulse cov: 2 ft: 2 corp: 1/1b lim: 4096 exec/s: 524288 rss: 481Mb

Running tests
Post Checkout code

```

Figure 6: Fuzzing in the CI solution

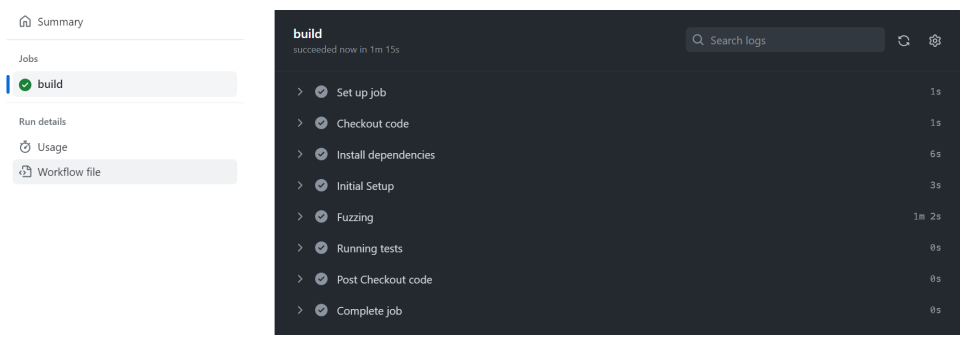


Figure 7: Final results in GitHub Actions