

# FUZZING ATTACK

**SUBJECT NAME:** Cryptography and Network Security

**SUBJECT CODE:** CS6008

**MODULE:** 04

**NAME:** R.Aadharsh

**REG.NO.:** 2019103604

**DATE:** 06/06/2022

**AIM:**

Using libfuzzer and AFL to fuzz your own C/C++ implementations, to implement a c/c++ code using fuzzer tools

**TOOLS INVOLVED:**

LIBFUZZER

AFL FUZZER

VISUAL STUDIO CODE

KALI LINUX TERMINAL (WSL)

WINDOWS (OPERATION SYSTEM)

**PROBLEM DESCRIPTION:**

Fuzzing is the process of attempting a large number of random inputs to code in order to find a flaw. You design a testbench for the code in question, link it with a fuzzing engine that creates random data, and run it on a server. If your testbench is sound, it will return with a set of inputs that will cause the code to crash hours, days, or weeks later. This procedure can be sped up by:

- Using a sanitizer: compiler-supported sanitizers add extra code to binaries to check for unlawful circumstances like out-of-bounds memory accesses that don't always result in a crash. As a result, the code under test is more likely to fail, and the fuzzer runs faster.
- Coverage-driven fuzzing: fuzzers can track the programme states reached by various inputs and direct the inputs in a way that tends to generate new ones.

Check up libFuzzer from the LLVM Subversion repository and build it according to their instructions. A test driver named LLVMFuzzerTestOneInput is provided as a function with C linkage. As a consequence, you'll have a standalone application that runs the code contained within that function. It monitors which pathways are exercised using instrumentation provided by the Clang compiler via the `-fsanitize-coverage` option, thus gcc isn't an option. To guarantee no initialization is omitted, it must be built using `-fsanitize=memory`. AFL is a stand-alone application that employs binary rewriting to instrument the code under test. Wrapper compilers are provided, which call any of the two compilers.

**INPUT:**

Getting input from a file

**OUTPUT:**

Given input string is to check whether it is a palindrome or not

## SCREENSHOTS:

### FUZZING WITH AFL:

#### Step 1:Installing AFL:

```
$ apt-get install -y afl  
$ apt-get source libxml2-utils
```

#### Step 2:

Next we configure libxml2 build to use AFL compilers and compile the xmllint utility.

```
$ cd libxml2/  
$ ./configure CC=afl-gcc CXX=afl-g++  
$ make xmllint
```

Step 3:Lastly we create a sample file with content "<a></a>" for AFL to start with and run the afl-fuzz.

```
$ echo "" > in/sample  
$ LD_LIBRARY_PATH=./.libs/ afl-fuzz -i ./in -o ./out -- ../libs/libxml2/xmllint -o /dev/null @@
```

Compiling and running the program using AFL

```
~/libxml2/fuzz on ◀ master! © 10:36:36  
$ afl-fuzz -i in -o out -- ../libs/xmllint -o /dev/null @@
```

```

afl-fuzz 2.43b by <lcamtuf@google.com>
[+] You have 4 CPU cores and 1 runnable tasks (utilization: 25%).
[+] Try parallel jobs - see /usr/local/share/doc/afl/parallel_fuzzing.txt.
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Setting up output directories...
[*] 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'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 8, map size = 1582, exec speed = 575 us
[+] All test cases processed.

[+] Here are some useful stats:

    Test case count : 1 favored, 0 variable, 1 total
    Bitmap range   : 1582 to 1582 bits (average: 1582.00 bits)
    Exec timing    : 575 to 575 us (average: 575 us)

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

```

american fuzzy lop 2.43b (xmllint)			
process timing		overall results	
run time : 0 days, 0 hrs, 0 min, 14 sec		cycles done : 0	
last new path : 0 days, 0 hrs, 0 min, 0 sec		total paths : 293	
last uniq crash : none seen yet		uniq crashes : 0	
last uniq hang : none seen yet		uniq hangs : 0	
cycle progress		map coverage	
now processing : 0 (0.00%)		map density : 2.41% / 5.02%	
paths timed out : 0 (0.00%)		count coverage : 2.27 bits/tuple	
stage progress		findings in depth	
now trying : havoc		favored paths : 1 (0.34%)	
stage execs : 22.1k/32.8k (67.46%)		new edges on : 165 (56.31%)	
total execs : 25.6k		total crashes : 0 (0 unique)	
exec speed : 1835/sec		total tmouts : 0 (0 unique)	
fuzzing strategy yields		path geometry	
bit flips : 24/64, 7/63, 8/61		levels : 2	
byte flips : 1/8, 3/7, 3/5		pending : 293	
arithmetics : 13/447, 0/25, 0/0		pend fav : 1	
known ints : 9/44, 18/196, 11/220		own finds : 292	
dictionary : 0/0, 0/0, 0/0		imported : n/a	
havoc : 0/0, 0/0		stability : 100.00%	
trim : 0.00%/1, 0.00%			
[cpu000: 50%]			

```
last uniq crash : none seen yet      uniq crashes : 0
last uniq hang  : none seen yet      uniq hangs   : 0
cycle progress
now processing  : 0 (0.00%)
paths timed out : 0 (0.00%)
stage progress
now trying      : havoc
stage execs     : 30.7k/32.8k (93.60%)
total execs     : 34.4k
exec speed      : 1834/sec
fuzzing strategy yields
bit flips       : 24/64, 7/63, 8/61
byte flips      : 1/8, 3/7, 3/5
arithmetics     : 13/447, 0/25, 0/0
known ints      : 9/44, 18/196, 11/220
dictionary      : 0/0, 0/0, 0/0
havoc           : 0/0, 0/0
trim            : 0.00%/1, 0.00%
map coverage
map density     : 2.41% / 5.13%
count coverage  : 2.29 bits/tuple
findings in depth
favored paths   : 1 (0.31%)
new edges on    : 179 (55.42%)
total crashes   : 0 (0 unique)
total tmouts    : 0 (0 unique)
path geometry
levels          : 2
pending         : 323
pend fav        : 1
own finds       : 322
imported        : n/a
stability       : 100.00%
^C [cpu000: 50%]

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

~/libxml2/fuzz on  master!  @ 10:37:33
$
```

Code:

```

1  #include <iostream>
2  #include <bits/stdc++.h>
3  #include <fstream>
4  using namespace std;
5  bool isPalindrome(string data, int s)
6  {
7      for (int i = 0; i < s / 2; i++)
8      {
9          if (data[i] != data[s - i - 1])
10         return false;
11     }
12     return true;
13 }
14 void parse_file(char *filename,vector<string>&input){
15     string i;
16     ifstream input_file(filename);
17     if(!input_file.is_open()){
18         cerr << "Could not open the file - '"<< filename << "'" << endl;
19         exit(-1);
20     }
21     while(getline(input_file,i)){
22         input.push_back(i);
23     }
24     input_file.close();
25 }
26 int main(int argc, char **argv)
27 {
28     string data;
29     if (argc < 2)
30     {
31         cout << "enter filename";
32         return -1;
33     }
34     vector<string>input;
35     parse_file(argv[1],input);
36     for(auto i : input){
37         if(isPalindrome(i,i.size())){
38             cout<<i<<"\t --> TRUE\n";
39         }
40         else{
41             cout<<i<<"\t --> FALSE\n";
42         }
43     }
44     return 0;
45 }

```

## Compiling and running with afl-g++

[illegible]

## LIBFUZZER

Let's now fuzz libxml2 with the LLVM libFuzzer. To start fuzzing, you'll first need to introduce a target function, `LLVMFuzzerTestOneInput`, that receives the fuzzed input buffer from libFuzzer. The code looks like this.

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data,
size_t Size) {
    DoSomethingInterestingWithMyAPI(Data, Size);
    return 0; // Non-zero return values are reserved for future use.
}
```



For fuzzing libxml2, Google's fuzzer test suite provides a good example which is as follows:

```
#include
#include
#include "libxml/xmlversion.h"
#include "libxml/parser.h"
#include "libxml/HTMLparser.h"
#include "libxml/tree.h"
void ignore (void * ctx, const char * msg, ...) {}
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t
size) {
    xmlSetGenericErrorFunc(NULL, &ignore);
    if (auto doc = xmlReadMemory(reinterpret_cast(data), size,
"noname.xml", NULL, 0))
        xmlFreeDoc(doc);
    return 0;
}
```

Before compiling our target function, we need to compile all dependencies with clang and `-fsanitize-coverage=trace-pc-guard`, to enable SanitizerCoverage coverage tracing. It is a good idea to also use `-fsanitize=address,undefined` in order to enable both the AddressSanitizer(ASAN) and the UndefinedBehaviorSanitizer(UBSAN) that catch many bugs that otherwise might be hard to find

```
$ git clone https://github.com/GNOME/libxml2 libxml2
$ cd libxml2
$ FUZZ_CXXFLAGS="-O2 -fno-omit-frame-pointer -g -fsanitize=address,undefined -fsanitize-coverage=trace-pc-guard"
$ ./autogen.sh
$ CXX="clang++-5.0 $FUZZ_CXXFLAGS" CC="clang-5.0 $FUZZ_CXXFLAGS"
CCLD="clang++-5.0 $FUZZ_CXXFLAGS" ./configure
$ make
```

The second step is to compile our target function, with the same flags, and link it with both the libFuzzer runtime and the libxml2 we compiled earlier.

```
$ clang++-5.0 -std=c++11 $FUZZ_CXXFLAGS -lFuzzer ./libxml-test.cc -I
./include ./libs/libxml2.a -lz -llzma -o libxml-fuzzer
```

Now we are ready to run our fuzzer

```
$ mkdir ./output
$ ./libxml-fuzzer ./output/
libfuzzer-demo:~$ ./libxml2/libxml-fuzzer ./output/
INFO: Seed: 2485440675
INFO: Loaded 1 modules (237370 guards): {0x13b3460, 0x149b140},
Loading corpus dir: ./output/
INFO: max_len is not provided, using 64
INFO: A corpus is not provided, starting from an empty corpus
#0  READ units: 1
#1  INITED cov: 2286 ft: 2087 corp: 1/1b exec/s: 0 rss: 51Mb
#2  NEW cov: 2286 ft: 2091 corp: 2/2b exec/s: 0 rss: 51Mb L: 1 MS: 1 ShuffleBytes-
#3  NEW cov: 2297 ft: 2317 corp: 3/3b exec/s: 0 rss: 51Mb L: 1 MS: 2 ShuffleBytes-ChangeByte-
#7  NEW cov: 2301 ft: 2427 corp: 4/5b exec/s: 0 rss: 51Mb L: 2 MS: 1 CopyPart-
#9  NEW cov: 2351 ft: 2486 corp: 5/9b exec/s: 0 rss: 52Mb L: 4 MS: 3 CopyPart-Change0InInt-CopyPart-
#10 NEW cov: 2386 ft: 2789 corp: 6/30b exec/s: 0 rss: 52Mb L: 29 MS: 4 CopyPart-Change0InInt-CopyPart-InsertRepeatedBytes-
#26 NEW cov: 2387 ft: 2791 corp: 7/70b exec/s: 0 rss: 52Mb L: 40 MS: 5 Change0InInt-CopyPart-InsertByte-InsertByte-CopyPart-
#32 NEW cov: 2391 ft: 2802 corp: 8/110b exec/s: 0 rss: 52Mb L: 30 MS: 1 InsertRepeatedBytes-
#33 NEW cov: 2391 ft: 2809 corp: 9/120b exec/s: 0 rss: 52Mb L: 4 MS: 2 InsertRepeatedBytes-CrossOver-
#36 NEW cov: 2391 ft: 2816 corp: 10/132b exec/s: 0 rss: 52Mb L: 12 MS: 5 InsertRepeatedBytes-CrossOver-CopyPart-Change0InInt-CMP- DE: "\xff\xff\xff\xff\xff\xff\xff-
#39 NEW cov: 2391 ft: 2823 corp: 11/150b exec/s: 0 rss: 52Mb L: 26 MS: 3 InsertByte-Change0InInt-InsertRepeatedBytes-
#65 NEW cov: 2391 ft: 2824 corp: 12/162b exec/s: 0 rss: 53Mb L: 4 MS: 4 ChangeByte-InsertByte-Change0InInt-CrossOver-
#66 NEW cov: 2391 ft: 2829 corp: 13/166b exec/s: 0 rss: 53Mb L: 4 MS: 5 ChangeByte-InsertByte-Change0InInt-CrossOver-Change0InInt-
#108 NEW cov: 2393 ft: 2831 corp: 14/226b exec/s: 0 rss: 54Mb L: 60 MS: 2 InsertRepeatedBytes-PersAutoDict- DE: "\xff\xff\xff\xff\xff\xff\xff-
#150 NEW cov: 2394 ft: 2832 corp: 15/230b exec/s: 0 rss: 55Mb L: 4 MS: 4 CMP-EraseBytes-ChangeByte-CrossOver- DE: "\x00\x00-
#337 NEW cov: 2394 ft: 2833 corp: 16/292b exec/s: 0 rss: 58Mb L: 62 MS: 1 InsertRepeatedBytes-
#412 NEW cov: 2394 ft: 2838 corp: 17/320b exec/s: 0 rss: 60Mb L: 28 MS: 1 InsertRepeatedBytes-
#647 NEW cov: 2394 ft: 2940 corp: 10/325b exec/s: 0 rss: 64Mb L: 5 MS: 1 CrossOver-
```

```

INFO: Seed: 2485440675
INFO: Loaded 1 modules (237370 guards): {0x13b3460, 0x149b140},
Loading corpus dir: ./output/
INFO: -max_len is not provided, using 64
INFO: A corpus is not provided, starting from an empty corpus
#0  READ units: 1
#1  INITED cov: 2206 ft: 2007 corp: 1/1b exec/s: 0 rss: 51Mb
#2  NEW cov: 2206 ft: 2091 corp: 2/2b exec/s: 0 rss: 51Mb L: 1 MS: 1 ShuffleBytes-
#3  NEW cov: 2297 ft: 2317 corp: 3/3b exec/s: 0 rss: 51Mb L: 1 MS: 2 ShuffleBytes-ChangeByte-
#7  NEW cov: 2301 ft: 2427 corp: 4/5b exec/s: 0 rss: 51Mb L: 2 MS: 1 CopyPart-
#9  NEW cov: 2351 ft: 2406 corp: 5/9b exec/s: 0 rss: 52Mb L: 4 MS: 3 CopyPart-ChangeBinInt-CopyPart-
#10 NEW cov: 2306 ft: 2709 corp: 6/30b exec/s: 0 rss: 52Mb L: 29 MS: 4 CopyPart-ChangeBinInt-CopyPart-InsertRepeatedBytes-
#26 NEW cov: 2307 ft: 2791 corp: 7/70b exec/s: 0 rss: 52Mb L: 40 MS: 5 ChangeBinInt-CopyPart-InsertByte-InsertByte-CopyPart-
#32 NEW cov: 2391 ft: 2802 corp: 8/116b exec/s: 0 rss: 52Mb L: 38 MS: 1 InsertRepeatedBytes-
#33 NEW cov: 2391 ft: 2809 corp: 9/120b exec/s: 0 rss: 52Mb L: 4 MS: 2 InsertRepeatedBytes-CrossOver-
#36 NEW cov: 2391 ft: 2816 corp: 10/132b exec/s: 0 rss: 52Mb L: 12 MS: 5 InsertRepeatedBytes-CrossOver-CopyPart-ChangeBinInt-CMP- DE: "\xff\xff\xff\xff\xff\xff\xff-
#39 NEW cov: 2391 ft: 2823 corp: 11/150b exec/s: 0 rss: 52Mb L: 26 MS: 3 InsertByte-ChangeBinInt-InsertRepeatedBytes-
#65 NEW cov: 2391 ft: 2824 corp: 12/162b exec/s: 0 rss: 53Mb L: 4 MS: 4 ChangeByte-InsertByte-ChangeBit-CrossOver-
#66 NEW cov: 2391 ft: 2829 corp: 13/166b exec/s: 0 rss: 53Mb L: 4 MS: 5 ChangeByte-InsertByte-ChangeBit-CrossOver-ChangeBinInt-
#190 NEW cov: 2393 ft: 2831 corp: 14/226b exec/s: 0 rss: 54Mb L: 60 MS: 2 InsertRepeatedBytes-PersAutoDict- DE: "\xff\xff\xff\xff\xff\xff\xff-
#150 NEW cov: 2394 ft: 2832 corp: 15/230b exec/s: 0 rss: 55Mb L: 4 MS: 4 CMP-EraseBytes-ChangeByte-CrossOver- DE: "\x00\x00"-
#337 NEW cov: 2394 ft: 2833 corp: 16/292b exec/s: 0 rss: 59Mb L: 62 MS: 1 InsertRepeatedBytes-
#412 NEW cov: 2394 ft: 2838 corp: 17/320b exec/s: 0 rss: 60Mb L: 38 MS: 1 InsertRepeatedBytes-
#647 NEW cov: 2394 ft: 2940 corp: 18/325b exec/s: 0 rss: 64Mb L: 5 MS: 1 CrossOver-
#776 NEW cov: 2395 ft: 2941 corp: 19/385b exec/s: 0 rss: 67Mb L: 60 MS: 2 CrossOver-InsertRepeatedBytes-
#782 NEW cov: 2395 ft: 3024 corp: 20/390b exec/s: 0 rss: 67Mb L: 5 MS: 1 CopyPart-
#1105 NEW cov: 2404 ft: 3033 corp: 21/391b exec/s: 0 rss: 73Mb L: 1 MS: 4 ShuffleBytes-ChangeByte-CopyPart-ChangeBit-
#1124 NEW cov: 2404 ft: 3034 corp: 22/390b exec/s: 0 rss: 74Mb L: 5 MS: 3 ChangeBit-PersAutoDict-ChangeBinInt- DE: "\x00\x00"-
#1352 NEW cov: 2404 ft: 3058 corp: 23/404b exec/s: 0 rss: 78Mb L: 8 MS: 1 CopyPart-
#1353 NEW cov: 2404 ft: 3063 corp: 24/413b exec/s: 0 rss: 78Mb L: 9 MS: 2 CopyPart-InsertByte-
#1582 NEW cov: 2695 ft: 3428 corp: 25/417b exec/s: 0 rss: 83Mb L: 4 MS: 1 ChangeBit-
#1583 NEW cov: 2696 ft: 3429 corp: 26/423b exec/s: 0 rss: 83Mb L: 6 MS: 2 ChangeBit-PersAutoDict- DE: "\x00\x00"-
#1589 NEW cov: 2696 ft: 3505 corp: 27/431b exec/s: 0 rss: 83Mb L: 8 MS: 3 CrossOver-CrossOver-InsertByte-
#1633 NEW cov: 2709 ft: 3540 corp: 28/430b exec/s: 0 rss: 84Mb L: 5 MS: 2 ChangeByte-InsertByte-
#1634 NEW cov: 2710 ft: 3576 corp: 29/482b exec/s: 0 rss: 84Mb L: 46 MS: 3 ChangeByte-InsertByte-InsertRepeatedBytes-
#1643 NEW cov: 2710 ft: 3588 corp: 30/489b exec/s: 0 rss: 84Mb L: 7 MS: 2 InsertByte-CopyPart-
#1707 NEW cov: 2740 ft: 3630 corp: 31/494b exec/s: 0 rss: 85Mb L: 5 MS: 1 CopyPart-
#1737 NEW cov: 2752 ft: 3634 corp: 32/540b exec/s: 0 rss: 86Mb L: 46 MS: 1 ChangeBinInt-
#1740 NEW cov: 2752 ft: 3638 corp: 33/590b exec/s: 0 rss: 86Mb L: 46 MS: 4 ChangeBinInt-ShuffleBytes-ChangeByte-CopyPart-
#1741 NEW cov: 2752 ft: 3652 corp: 34/637b exec/s: 0 rss: 86Mb L: 51 MS: 5 ChangeBinInt-ShuffleBytes-ChangeByte-CopyPart-InsertRepeatedBytes-
#1742 NEW cov: 2752 ft: 3664 corp: 35/649b exec/s: 0 rss: 86Mb L: 12 MS: 1 PersAutoDict- DE: "\xff\xff\xff\xff\xff\xff\xff-
#1743 NEW cov: 2752 ft: 3676 corp: 36/673b exec/s: 0 rss: 86Mb L: 24 MS: 2 PersAutoDict-InsertRepeatedBytes- DE: "\xff\xff\xff\xff\xff\xff\xff-

```

Code:

```

#include <stdint.h>
#include <stddef.h>
bool fuzz(const uint8_t *data, size_t s){
    for(int i = 0 ; i <= s/2 ; i++){
        if(data[i] != data[s-i-1])
            return false;
    }
    return true;
}

extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t s){
    fuzz(data,s);
    return 0;
}

```

When the value of I is equal to half of the value of size in the for loop of the function "fuzz," an error will occur since there is no condition for palindrome. To include libfuzzer in our software, we utilised Clang+. The libfuzzer driver LLVMFuzzerTestOneInput

is used to test the fuzz() function.  
We'll compile the results and see what occurs when Max\_len = 20 with max ip size of 5. clang++ -g -

```
$ clang++ -g -fsanitize=address, fuzzer libfuzz.cc -o fuzz
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 360876546
INFO: Loaded 1 modules (7 inline 8-bit counters): 7 [0x54bee0, 0x54bee7],
INFO: Loaded 1 PC tables (7 PCs): 7 [0x524070, 0x5240e0],
=====
==384==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000002f at pc 0x00000050cc52 bp 0x7ffde865af70 sp 0x7ffde865af68
READ of size 1 at 0x60200000002f thread T0
#0 0x50cc51 in fuzz(unsigned char const*, unsigned long) /mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/libfuzz.cc:6:23
#1 0x50cd84 in LLVMFuzzerTestOneInput /mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/libfuzz.cc:13:5
#2 0x4089b3 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x4408b3)
#3 0x441f20 in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >&) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x441f20)
#4 0x442482 in fuzzer::Fuzzer::Loop(std::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >&) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x442482)
#5 0x42f92d in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x42f92d)
#6 0x45b6a2 in main (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x45b6a2)
#7 0x7fcd95f007fc in __libc_start_main csu/../csu/libc-start.c:332:16
#8 0x4238d9 in _start (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x4238d9)

0x60200000002f is located 1 bytes to the left of 1-byte region [0x602000000030,0x602000000031]
allocated by thread T0 here:
#0 0x50a37d in operator new[](unsigned long) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x50a37d)
#1 0x4408c2 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x4408c2)
#2 0x441f20 in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >&) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x441f20)
#3 0x442482 in fuzzer::Fuzzer::Loop(std::vector<fuzzer::SizedFile, fuzzer::fuzzer_allocator<fuzzer::SizedFile> >&) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x442482)
#4 0x42f92d in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x42f92d)
#5 0x45b6a2 in main (/mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/fuzz+0x45b6a2)
#6 0x7fcd95f007fc in __libc_start_main csu/../csu/libc-start.c:332:16

SUMMARY: AddressSanitizer: heap-buffer-overflow /mnt/e/clg 6th sem/crypto&net security/assignment/fuzz/trial/libfuzz.cc:6:23 in fuzz(unsigned char const*, unsigned long)
```

Here the error says that heap-overflow has occurred at address 0x60200000002f at instruction pc

0x0000000050xx52 bp 0x7ffde865af70 bp 0x7ffde865af68.

Program after removal of discovered bug:  
Code:

```

#include <stdint.h>
#include <stddef.h>
#include <signal.h>
bool isPalindrome(const uint8_t *data, size_t s){
    for(int i = 0 ; i < s/2 ; i++){
        if(data[i] != data[s-i-1])
            return false;
    }
    return true;
}
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t s){
    isPalindrome(data,s);
    return 0;
}

```

```

$ ./vuln -max_len=3
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 3727234840
INFO: Loaded 1 modules (7 inline 8-bit counters): 7 [0x54bee0, 0x54bee7),
INFO: Loaded 1 PC tables (7 PCs): 7 [0x524070,0x5240e0),
INFO: A corpus is not provided, starting from an empty corpus
#2      INITED cov: 4 ft: 4 corp: 1/1b exec/s: 0 rss: 29Mb
#6      NEW    cov: 5 ft: 5 corp: 2/3b lim: 3 exec/s: 0 rss: 30Mb L: 2/2 MS: 4 ShuffleBytes-ChangeBit-ChangeByte-CopyPar
t-
#7      NEW    cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 0 rss: 30Mb L: 2/2 MS: 1 InsertByte-
#4194304      pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1398101 rss: 217Mb
#8388608      pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1398101 rss: 403Mb
#16777216     pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1290555 rss: 775Mb
#33554432     pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1118481 rss: 1159Mb
#67108864     pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1100145 rss: 1159Mb
#134217728    pulse cov: 7 ft: 7 corp: 3/5b lim: 3 exec/s: 1109237 rss: 1159Mb

```

Crash reported at when the input string is null

```

trial > crash-da39a3ee5e6b4b0d3255bfef95601890afd80709
1

```