



## Zint Barcode Generator Tickets

A barcode encoding library supporting over 50 symbologies.

Brought to you by: g3rrk, gitlost, oehhar, schoepe, sdanig

### #218 Stack Buffer Overflow in EAN Generator



Milestone: [1.0](#) Status: closed Owner: nobody Labels: None  
Updated: 2021-08-14 Created: 2021-02-25 Creator: [Jan Schrewe](#) Private: No

Hi,  
I used [CI-Fuzz](#) to fuzz the EAN Generator. Last year Chritian Hartlage already found multiple [bugs](#) in a previous version of zint this way.  
I discovered a stack buffer overflow in the EAN Generator in the currrent release 2.9.1.  
The bug can be triggered by using the input "000002000000200+203" with the reproducer described in the ticket by Christian linked above.

This is the stack trace (compiled with address sanitizer)

```
==14==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffe43a3dcc5 at pc 0x000000484109 bp
0x7ffe43a3d1b0 sp 0x7ffe43a3c950
WRITE of size 6 at 0x7ffe43a3dcc5 thread T0
#0 0x484108 in strcat /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/asan/asan_interceptors.cpp:390:7
#1 0x7f9a57edb02a in ean_leading_zeros /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/upcean.c:676:9
#2 0x7f9a57edc057 in eanx /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/upcean.c:729:5
#3 0x7f9a57dde4f3 in reduced_charset /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/library.c:790:28
#4 0x7f9a57dd170d in extended_or_reduced_charset /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/library.c:736:33
#5 0x7f9a57dc7ac2 in ZBarcode_Encode /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/library.c:1324:20
#6 0x4cb57f in LLVMFuzzerTestOneInput /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/code-intelligence/fuzz_targets/ean_fuzzer.cpp:23:3
#7 0x504501 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const, unsigned long) /llvmbuild/llvm-project-llvmorg-
11.0.0/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:560:15
#8 0x503c45 in fuzzer::Fuzzer::RunOne(unsigned char const, unsigned long, bool, fuzzer::InputInfo, bool) /llvmbuild/llvm-
project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:472:3
#9 0x505670 in fuzzer::Fuzzer::MutateAndTestOne() /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-
rt/lib/fuzzer/FuzzerLoop.cpp:703:19
#10 0x5060e5 in fuzzer::Fuzzer::Loop(std::__Fuzzer::vector<fuzzer::sizedfile, fuzzer::fuzzer_allocator<fuzzer::sizedfile>"">
> &) /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:839:5
#11 0x4f5ae5 in fuzzer::FuzzerDriver(int, char, int (*)(unsigned char const, unsigned long)) /llvmbuild/llvm-project-llvmorg-
11.0.0/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:847:6
#12 0x51d8c2 in main /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10
#13 0x7f9a575460b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
#14 0x41ef3d in _start (/projects/zint-006bf471/libfuzzer/address/fuzz_target_ean_fuzzer+0x41ef3d)</fuzzer::sizedfile,>
```

Address 0x7ffe43a3dcc5 is located in stack of thread T0 at offset 1285 in frame  
#0 0x7f9a57edb1df in eanx /home/user/local/share/code-intelligence/projects/zint-
006bf471/libfuzzer/address/backend/upcean.c:685

This frame has 5 object(s):  
[32, 52) 'first\_part' (line 686)  
[96, 103) 'second\_part' (line 686)  
[128, 1128) 'dest' (line 686)  
[1264, 1285) 'local\_source' (line 687) <== Memory access at offset 1285 overflows this variable  
[1328, 1332) 'with\_addon' (line 689)  
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork  
(longjmp and C++ exceptions are supported)

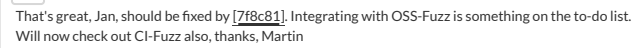
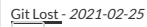
SUMMARY: AddressSanitizer: stack-buffer-overflow /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-
rt/lib/asan/asan\_interceptors.cpp:390:7 in strcat  
Shadow bytes around the buggy address:

```
0x10004873fb40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fb50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fb60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fb70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fb80: 00 00 00 00 00 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
=>0x10004873fb90: f2 f2 f2 f2 f2 00 00[05]f2 f2 f2 f2 04 f3
0x10004873fba0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fbb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fbc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fbd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10004873fbe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

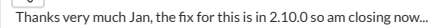
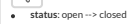
Shadow byte legend (one shadow byte represents 8 application bytes):

Addressable: 00  
Partially addressable: 01 02 03 04 05 06 07  
Heap left redzone: fa  
Freed heap region: fd  
Stack left redzone: f1  
Stack mid redzone: f2  
Stack right redzone: f3  
Stack after return: f5  
Stack use after scope: f8  
Global redzone: f9  
Global init order: f6  
Poisoned by user: f7

## Discussion



Commit: [7f8c81]



SourceForge

### Top Downloaded Projects

+1 (858) 454-5900

## Site Status



Advertise