Zint Barcode Generator Tickets

A barcode encoding library supporting over 50 symbologies.
Brought to you by: g3rrk, gitlost, oehhar, schoepe, sdanig

#232 Global buffer overflow in rs_encode_uint function

Milestone: 1.0 **Status:** closed **Owner:** nobody **Labels:** None
Updated: 2021-08-14 **Created:** 2021-07-05 **Creator:** Jan Schrewe **Private:** No

Hello,

at Code Intelligence we discovered a bug in the barcode parser using our fuzzing tool CI-Fuzz. It is a global buffer overflow in the latest version of zint which should be considered a security vulnerability. Below you can find all the details about the finding.

If you need additional information about the finding that I might have forgotten to include or if I can support you in any other way, please let me know.

Regards,
Jan Schrewe

Code of the used fuzz target:
https://github.com/ci-fuzz/zint/blob/master/.code-intelligence/fuzz_targets/codeone_fuzzer.cpp

Crashing Input:
{{-06755712162106130000000829203983ÿ

==15==ERROR: AddressSanitizer: global-buffer-overflow on address 0x7f151a1d4a7a at pc 0x7f1519cc1667 bp 0x7fff1f76db10 sp 0x7fff1f76db08 READ of size 1 at 0x7f151a1d4a7a thread T0 #0 0x7f1519cc1666 in rs_encode_uint /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/reedsol.c:170:34 #1 0x7f151a0aac73 in code_one /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/code1.c:1154:13 #2 0x7f1519cb0747 in reduced_charset /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/library.c:903:46 #3 0x7f1519ca1e7d in extended_or_reduced_charset /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/library.c:737:33 #4 0x7f1519c985e8 in ZBarcode_Encode /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/library.c:1311:20 #5 0x4cb5af in LLVMFuzzerTestOneInput /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/.code-intelligence/fuzz_targets/codeone_fuzzer.cpp:21:3 #6 0x504531 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const, *unsigned long) /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:560:15 #7 0x503c75 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 #8 0x505c17 in fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::__Fuzzer::vector<fuzzer::sizedfile, fuzzer::fuzzer_allocator<fuzzer::sizedfile=""> >&) /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:772:7 #9 0x505e19 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:801:3 #10 0x4f5b15 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 #11 0x51d8f2 in main /llvmbuild/llvm-project-llvmorg-11.0.0/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10 #12 0x7f15193dc0b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2) #13 0x41ef3d in _start (/projects/zint-code-d7410844/libfuzzer/address/fuzz_target_codeone_fuzzer+0x41ef3d) 0x7f151a1d4a7a is located 38 bytes to the left of global variable 'alog_0x12d' defined in '/home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/reedsol_logs.h:176:28' (0x7f151a1d4aa0) of size 510 0x7f151a1d4a7a is located 26 bytes to the right of global variable 'logt_0x12d' defined in '/home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/reedsol_logs.h:158:28' (0x7f151a1d4960) of size 256 SUMMARY: AddressSanitizer: global-buffer-overflow /home/user/.local/share/code-intelligence/projects/zint-code-d7410844/libfuzzer/address/backend/reedsol.c:170:34 in rs_encode_uint Shadow bytes around the buggy address: 0x0fe3234328f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 0x0fe323432920: f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 00 00 00 00 00 0x0fe323432930: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 =>0x0fe323432940: 00 00 00 00 00 00 00 00 00 00 00 00 00 f9 f9 f9[f9] 0x0fe323432950: f9 f9 f9 f9 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432960: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432970: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432980: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0x0fe323432990: 00 00 00 06 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 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 Container overflow: fc Array cookie: ac Intra object redzone: bb ASan internal: fe Left alloca redzone: ca Right alloca redzone: cb Shadow gap: cc ==15==ABORTING MS: 0 ; base unit: 0000000000000000000000000000000000000000 0x7b,0x7b,0x2d,0x30,0x36,0x14,0x37,0x35,0x35,0x37,0x31,0x32,0x31,0x36,0x32,0x31,0x30,0x36,0x31,0x33,0x30,0x30,0x30,0x3
{{-06\x14755712162106130000000829203983\xff artifact_prefix='./'; Test unit written to ./crash-face156ac55c1d630489f22a9a4a686496f4fab3 Base64: e3stMDYUNzU1NzEyMTYyMTA2MTMwMDAwMDAwODI5MjAzOTgz/w==</fuzzer::sizedfile,></fuzzer::sizedfile,>

Discussion

Git Lost - *2021-07-05*

Thanks very much Jan, this should be fixed by commit [9b02cd] where `is_last_single_ascii()` was indexing by `sp + 1` after checking that `sp` was pointing to the last char, doh.

I note that you're fuzzing for 30 minutes, and on individual symbologies, which is much more fuzzing than anything I've done. Will keep that in mind on my own testing.

Much appreciated, Martin

Related