Zint Barcode Generator Tickets

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

#181 Segfault in EAN Generator

Milestone: 2.0 **Status:** closed **Owner:** Harald Oehlmann **Labels:** bug (2)
Updated: 2020-04-01 **Created:** 2020-02-13 **Creator:** Christian Hartlage **Private:** No

Hi,
I set up fuzzing for the barcode generation and found a few bugs in different barcode generations. This Ticket is focussing on the EAN creation, because only a few ASCII characters input are enough to trigger the bug.

You can either input "55++15" into the qt-ui or use my reproducer.

This little reprocuder triggers the bug:

```
#include <zint.h>
#include <string>

int main() {
  std::string input = "55++15";
  struct zint_symbol *my_symbol;my_symbol = ZBarcode_Create();

  my_symbol->symbology = BARCODE_EANX;
  ZBarcode_Encode_and_Buffer(my_symbol, (unsigned char *) input.c_str(), input.size() , 0);

  ZBarcode_Delete(my_symbol);

  return 0;
}
```

Link against the current libzint.so as generated by cmake and the segfault will occur.
When compiled with address sanitizer, the following stacktrace will be printed:

```
/home/c/.local/share/code-intelligence/projects/project-4905718a-4da1-11ea-bbe0-8c1645a161c
SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior /home/c/.local/share/code-intellige
/home/c/.local/share/code-intelligence/projects/project-4905718a-4da1-11ea-bbe0-8c1645a161c
/usr/include/string.h:122:14: note: nonnull attribute specified here
SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior /home/c/.local/share/code-intellige
AddressSanitizer:DEADLYSIGNAL
=================================================================
==18211==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x7f77850ea5a1
==18211==The signal is caused by a READ memory access.
==18211==Hint: address points to the zero page.
    #0 0x7f77850ea5a0  /build/glibc-OTsEL5/glibc-2.27/string/../sysdeps/x86_64/multiarch/st
    #1 0x47edaf in strcpy /tmp/final/llvm.src/projects/compiler-rt/lib/asan/asan_intercepto
    #2 0x7f7786599893 in add_on /home/c/.local/share/code-intelligence/projects/project-490
    #3 0x7f77865971cb in eanx /home/c/.local/share/code-intelligence/projects/project-49057
    #4 0x7f7786554046 in reduced_charset /home/c/.local/share/code-intelligence/projects/pr
    #5 0x7f7786549e1b in ZBarcode_Encode /home/c/.local/share/code-intelligence/projects/pr
    #6 0x7f778655228f in ZBarcode_Encode_and_Buffer /home/c/.local/share/code-intelligence/p
    #7 0x4c57c2 in main (/home/c/.local/share/code-intelligence/projects/project-4905718a-4
    #8 0x7f7784f7db96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-s
    #9 0x41b6b9 in _start (/home/c/.local/share/code-intelligence/projects/project-4905718a
```

I will create bug reports for the other barcodes when I have some time in the next days. In order to trigger them the input needs to contain some non-printable characters so I don't think they are as significant as the EAN one.

Affected barcodes are AUSPOST family, CODEONE, DOTCODE, VIN and CODABLOCKF

Best regards,

Chris

Discussion

Robin Stuart - *2020-02-15*

Hi Chris,

Thank you for finding this and taking the time to let us know. I have added a check to the UPC/EAN code to throw an error if more than one + character is included in the input data.

I look forward to seeing what else you find!

Robin.

Log in to add a comment

Christian Hartlage - *2020-02-16*

Hi Robin,

Another bug, that I found in the Australian Post family of barcodes can be triggered when inputting the following byte sequence: `0x41 0x00 0x00 0x00`, an uppercase A followed by 3 null bytes. This will lead to a heap buffer read overflow.

If you compile it with address sanitizer (-DZINT_SANITIZE=TRUE). All line numbers are from zint-2.7.1.

This bug is strange I can trigger it only if I compile zint with gcc (version 9.3.0) and asan, with clang and

telepen.c - telepen()

Best regards,

Chris

DOTCODE crashes due to a read dynamic-stack-buffer-overflow with input: 0x28, 0x87, 0x27

```
==831668==ERROR: AddressSanitizer: dynamic-stack-buffer-overflow on address
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
==831668==ABORTING
```

All other bugs are fixed in my opinion, thanks for the replay and your work done here.

In *telepen.c* `63 char dest[512]; /*14 + 30 * 14 + 14 + 14 + 1 ~ 512 */` the *dest* array is too small because the biggest element in `41 static char *TeleTable[]` is 16 byte.

I think the best solution is to increase the *dest[]* buffer. (If I calculate it correct `111 char dest[1024];` should be large enough)

dotcode.tar.gz

library.c - ZBarcode_Encode()

```
./zint -i ZBarcode_Encode

=================================================================
==169091==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200
READ of size 2 at 0x602000000011 thread T0
    #0 0x7fd5f40d02e0 in __interceptor_strlen /build/gcc/src/gcc/libsaniti
    #1 0x560f4ea5d8d7 in ZBarcode_Encode /zint-2.7.1/backend/library.c:995
    #2 0x560f4ea63d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #3 0x560f4ea53127 in main /zint-2.7.1/frontend/main.c:733
    #4 0x7fd5f33c8022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #5 0x560f4ea5598d in _start (/zint-2.7.1/build-asan/frontend/zint+0x29

0x602000000011 is located 0 bytes to the right of 1-byte region [0x6020000
allocated by thread T0 here:
    #0 0x7fd5f4178b3a in __interceptor_malloc /build/gcc/src/gcc/libsaniti
    #1 0x560f4ea63c22 in ZBarcode_Encode_File /zint-2.7.1/backend/library.

SUMMARY: AddressSanitizer: heap-buffer-overflow /build/gcc/src/gcc/libsani
Shadow bytes around the buggy address:
  0x0c047fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c047fff8000: fa fa[01]fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
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
==169091==ABORTING
```

If you execute zint with an empty file you get a valid buffer of size zero in *library.c* `1457 buffer = (unsigned char *) malloc(fileLen * sizeof (unsigned char));` and then you call `995 in_length = (int)ustrlen(source);` on that buffer of size zero. Maybe you can return a *ZINT_ERROR* after `1444 fileLen = ftell(file);` if the file size is zero.

maxicode.c - maxi_text_process()

```
./zint -b57 -i maxicode-b57

/zint-2.7.1/backend/maxicode.c:433:41: runtime error: index 144 out of bou
/zint-2.7.1/backend/maxicode.c:433:41: runtime error: load of address 0x7f
0x7ffdce6f7020: note: pointer points here
 3b 00 00 00  50 70 6f ce fd 7f 00 00  1e 00 00 00 00 00 00 00  21 37 7c 0
             ^
==========================================================
==169991==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffc
READ of size 4 at 0x7ffdce6f7020 thread T0
    #0 0x5578d1a00da8 in maxi_text_process /zint-2.7.1/backend/maxicode.c:
    #1 0x5578d1a07027 in maxicode/zint-2.7.1/backend/maxicode.c:679
    #2 0x5578d190da20 in reduced_charset /zint-2.7.1/backend/library.c:857
    #3 0x5578d190da20 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #4 0x5578d19139b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #5 0x5578d1917d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #6 0x5578d1907127 in main /zint-2.7.1/frontend/main.c:733
    #7 0x7f8b0aa13022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #8 0x5578d190998d in _start (/zint-2.7.1/build-asan/frontend/zint+0x29

Address 0x7ffdce6f7020 is located in stack of thread T0 at offset 1440 in
    #0 0x5578d19f856f in maxi_text_process /zint-2.7.1/backend/maxicode.c:

  This frame has 6 object(s):
    [32, 40) 'set15' (line 157)
    [64, 72) 'set12' (line 158)
    [96, 116) 'set12345' (line 159)
    [160, 736) 'set' (line 155)
    [864, 1440) 'character' (line 155) <== Memory access at offset 1440 ov
    [1568, 1579) 'substring' (line 429)
HINT: this may be a false positive if your program uses some custom stack
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /zint-2.7.1/backend/maxic
Shadow bytes around the buggy address:
  0x100039cd6db0: f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 00 00 00 00
  0x100039cd6dc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100039cd6dd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100039cd6de0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100039cd6df0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100039cd6e00: 00 00 00 00[f2]f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
  0x100039cd6e10: f2 f2 f2 f2 00 03 f3 f3 00 00 00 00 00 00 00 00
  0x100039cd6e20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100039cd6e30: 00 00 00 00 f1 f1 f1 f1 00 00 00 04 f2 f2 f2 f2
  0x100039cd6e40: 04 f2 04 f2 00 04 f2 f2 00 07 f2 f2 00 07 f2 f2
  0x100039cd6e50: 00 00 00 06 f2 f2 f2 f2 00 00 00 06 f2 f2 f2 f2
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
==169991==ABORTING
```

I think the problem is this part of the code in *maxicode.c*

```
150 static int maxi_text_process(int mode, unsigned char source[], int
151     /* This code doesn't make use of [Lock in C], [Lock in D]
152     and [Lock in E] and so is not always the most efficient at
153     compressing data, but should suffice for most applications */
154
155     int set[144], character[144], i, j, done, count, current_set;

  ...

443             character[i + 5] = (value & 0x3f);
444
445             i += 6;
446             for (j = i; j < 140; j++) {
447                 set[j] = set[j + 3];
448                 character[j] = character[j + 3];
449             }
450             length -= 3;
451         } else {
452             i++;
453         }
454     } while (i <= 143);
```

in line *454* you check `i <= 143` but in line *443* you access `character[i + 5]` this overflows if `i >= 139`

code1.c - c1_encode()

```
./zint -b141 -i c1_encode-b141

=================================================================
==170204==ERROR: AddressSanitizer: dynamic-stack-buffer-overflow on addres
READ of size 1 at 0x7ffdac05ffe7 thread T0
    #0 0x55b0b56d03d1 in c1_encode /zint-2.7.1/backend/code1.c:445
    #1 0x55b0b56d1f24 in code_one /zint-2.7.1/backend/code1.c:1414
    #2 0x55b0b558d5d4 in reduced_charset /zint-2.7.1/backend/library.c:847
    #3 0x55b0b558d5d4 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #4 0x55b0b559117e in ZBarcode_Encode /zint-2.7.1/backend/library.c:120
    #5 0x55b0b5597d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #6 0x55b0b5587127 in main /zint-2.7.1/frontend/main.c:733
    #7 0x7f983dfa1022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #8 0x55b0b558998d in _start (/zint-2.7.1/build-asan/frontend/zint+0x29

Address 0x7ffdac05ffe7 is located in stack of thread T0 at offset 36855 in
    #0 0x55b0b56d1a2f in code_one /zint-2.7.1/backend/code1.c:1187

  This frame has 13 object(s):
    [48, 108) 'data' (line 1203)
    [144, 204) 'ecc' (line 1203)
    [240, 340) 'ecc' (line 1307)
    [384, 504) 'stream' (line 1204)
    [544, 704) 'data' (line 1307)
    [768, 992) 'elreg' (line 1202)
    [1056, 1316) 'stream' (line 1308)
    [1392, 1692) 'sub_ecc' (line 1406)
    [1760, 2520) 'sub_data' (line 1406)
    [2656, 5056) 'ecc' (line 1405)
    [5184, 11184) 'data' (line 1405)
    [11440, 19840) 'stream' (line 1407)
    [20096, 36416) 'datagrid' (line 1190) <== Memory access at offset 3685
HINT: this may be a false positive if your program uses some custom stack
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: dynamic-stack-buffer-overflow /zint-2.7.1/backe
Shadow bytes around the buggy address:
  0x100035803fa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100035803fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100035803fc0: 00 00 00 00 00 00 00 f3 f3 f3 f3 f3 f3 f3 f3 f3
  0x100035803fd0: f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3 f3
  0x100035803fe0: f3 f3 f3 f3 f3 f3 f3 00 00 00 00 00 00 00 00 00
=>0x100035803ff0: 00 00 00 00 ca ca ca ca 00 00 00 05[cb]cb cb cb
  0x100035804000: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
  0x100035804010: 00 00 00 f2 f2 f2 02 f2 00 00 00 00 00 00 00 00
  0x100035804020: 00 00 00 00 00 00 01 f3 f3 f3 f3 f3 00 00 00 00
  0x100035804030: 00 00 00 00 00 00 00 00 ca ca ca ca 00 00 00 05
  0x100035804040: cb cb cb cb 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
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==170204==ABORTING
```

this error occurs in the *do while* loop in *c1_encode()*

```
353  static int c1_encode()
...
388      do {
...
1020     } while (sp < length);
```

the check of `sp < length` in line *1020* is not enough because there are lot of `source[sp + i]` (and *source[sp + 1|2]*) accesses which will overflow if `sp + i >= length`

aztec.c - aztec_text_process()

```
./zint -b92 -i aztec_text_process-leak-b92
Error 502: Input too long or too many extended ASCII characters

=================================================================
==170410==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 2252 byte(s) in 1 object(s) allocated from:
    #0 0x7f4ba02bfb3a in __interceptor_malloc /build/gcc/src/gcc/libsaniti
    #1 0x5588315999be in aztec_text_process /zint-2.7.1/backend/aztec.c:14
    #2 0x5588315999be in aztec /zint-2.7.1/backend/aztec.c:1007
    #3 0x558831486a79 in reduced_charset /zint-2.7.1/backend/library.c:859
    #4 0x558831486a79 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #5 0x55883148c9b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #6 0x558831490d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.

Direct leak of 2252 byte(s) in 1 object(s) allocated from:
    #0 0x7f4ba02bfb3a in __interceptor_malloc /build/gcc/src/gcc/libsaniti
    #1 0x5588315999d0 in aztec_text_process /zint-2.7.1/backend/aztec.c:14
    #2 0x5588315999d0 in aztec /zint-2.7.1/backend/aztec.c:1007
    #3 0x558831486a79 in reduced_charset /zint-2.7.1/backend/library.c:859
    #4 0x558831486a79 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #5 0x55883148c9b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #6 0x558831490d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.

Direct leak of 2252 byte(s) in 1 object(s) allocated from:
    #0 0x7f4ba02bfb3a in __interceptor_malloc /build/gcc/src/gcc/libsaniti
    #1 0x5588315999ac in aztec_text_process /zint-2.7.1/backend/aztec.c:13
    #2 0x5588315999ac in aztec /zint-2.7.1/backend/aztec.c:1007
    #3 0x558831486a79 in reduced_charset /zint-2.7.1/backend/library.c:859
    #4 0x558831486a79 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #5 0x55883148c9b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #6 0x558831490d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.

SUMMARY: AddressSanitizer: 6756 byte(s) leaked in 3 allocation(s).
```

in *aztec.c* the memory allocated at lines *139-141*

```
    139     encode_mode=(char*)malloc(src_len + 1);
    140     reduced_source=(unsigned char*)malloc(src_len + 1);
    141     reduced_encode_mode=(char*)malloc(src_len + 1);
```

is not freed if you return at line *724* this should be easy to fix, i think you can free all three buffers before line *724*

```
    723                 if (count > 2079) {
    724                     return ZINT_ERROR_TOO_LONG;
    725                 }
```

library.c - ZBarcode_Encode_File()

```
./zint -i .
Is a directory

=================================================================
==170571==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 82 byte(s) in 1 object(s) allocated from:
    #0 0x7efebabc7b3a in __interceptor_malloc /build/gcc/src/gcc/libsaniti
    #1 0x5639f9177c22 in ZBarcode_Encode_File /zint-2.7.1/backend/library.

SUMMARY: AddressSanitizer: 82 byte(s) leaked in 1 allocation(s).
```

here its the same again the *buffer* allocated at line *1454* is not freed if you return at line *1469*

```
    1454    buffer = (unsigned char *) malloc(fileLen * sizeof (unsigned
    1465    do {
    1466        n = fread(buffer + nRead, 1, fileLen - nRead, file);
    1467        if (ferror(file)) {
    1468            strcpy(symbol->errtxt, strerror(errno));
    1469            return ZINT_ERROR_INVALID_DATA;
    1470        }
    1471        nRead += n;
    1472    } while (!feof(file) && (0 < n) && (nRead < fileLen));
```

i think the same can happen at line *1472*

dotcode.c

```
./zint -b115 -i make_dotstream-b115

/zint-2.7.1/backend/dotcode.c:1096:32: runtime error: index 233 out of bou
/zint-2.7.1/backend/dotcode.c:1096:32: runtime error: load of address 0x55
0x55d06c9f3192: note: pointer points here
 67 2f  7a 69 6e 74 2d 72 65 70  6f 72 74 2f 7a 69 6e 74  2d 32 2e 37 2e 3
```

```
1086 static size_t make_dotstream(unsigned char masked_array[], int array_
1087     int i;
1088
1089     dot_stream[0] = '\0';
1090
1091     /* Mask value is encoded as two dots */
1092     bin_append(masked_array[0], 2, dot_stream);
1093
1094     /* The rest of the data uses 9-bit dot patterns from Annex C */
1095     for (i = 1; i < array_length; i++) {
1096         bin_append(dot_patterns[masked_array[i]], 9, dot_stream);
1097     }
1098
1099     return strlen(dot_stream);
1100 }
```

in line 1096 if masked_array[i] is >= 113 memory is read outside of dot_patterns[113]

```
1212 static void apply_mask(int mask, int data_length, unsigned char *mask
1213     int weight = 0;
1214     int j;
1215
1216     switch (mask) {
1217         case 0:
1218             masked_codeword_array[0] = 0;
1219             for (j = 0; j < data_length; j++) {
1220                 masked_codeword_array[j + 1] = codeword_array[j];
```

```
1269 INTERNAL int dotcode(struct zint_symbol *symbol, const unsigned char
...
1298     data_length = dotcode_encode_message(symbol, source, length, code
...
1423         apply_mask(i, data_length, masked_codeword_array, codeword_ar
1424
1425         dot_stream_length = make_dotstream(masked_codeword_array, (da
```

codeword_array is generated by *dotcode_encode_message()* at line 1298 in *dotcode()*

masked_codeword_array is generated by *apply_mask()* at line 1423 in *dotcode()*

If the *codeword_array* should not contain values >= 113 the error is in *dotcode_encode_message()* (poc creates codeword_array[395] == 233)

If the *codeword_array* can contain values >= 113 the error is in *apply_mask()* at 1220 => asked_codeword_array[j + 1] = codeword_array[j] % 113;

pdf417.c - pdf417() / micro_pdf417()

```
./zint -b55 -i pdf417-micro_pdf417-b55-b84

/zint-2.7.1/backend/pdf417.c:572:17: runtime error: index 1000 out of bour
/zint-2.7.1/backend/pdf417.c:572:30: runtime error: store to address 0x55b
0x55b511796d60: note: pointer points here
 85 03 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 0
              ^
==========================================================
==171185==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55b
WRITE of size 4 at 0x55b511796d60 thread T0
    #0 0x55b5113443ae in pdf417 /zint-2.7.1/backend/pdf417.c:572
    #1 0x55b511344a78 in pdf417enc /zint-2.7.1/backend/pdf417.c:858
    #2 0x55b5112893ae in reduced_charset /zint-2.7.1/backend/library.c:853
    #3 0x55b5112893ae in extended_or_reduced_charset /zint-2.7.1/backend/l
    #4 0x55b51128f9b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #5 0x55b511293d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #6 0x55b511283127 in main /zint-2.7.1/frontend/main.c:733
    #7 0x7f1f578ad022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #8 0x55b51128598d in _start (/zint-2.7.1/build-asan/frontend/zint+0x29

0x55b511796d60 is located 32 bytes to the left of global variable 'maxi_co
0x55b511796d60 is located 0 bytes to the right of global variable 'liste'
SUMMARY: AddressSanitizer: global-buffer-overflow /zint-2.7.1/backend/pdf4
Shadow bytes around the buggy address:
  0x0ab7222ead50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222ead60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222ead70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222ead80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222ead90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0ab7222eada0: 00 00 00 00 00 00 00 00 00 00 00 00 00[f9]f9 f9 f9
  0x0ab7222eadb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222eadc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222eadd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222eade0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab7222eadf0: 00 00 00 00 00 00 00 00 f9 f9 f9 f9 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
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==171185==ABORTING
```

this bug happens with `./zint -b55` (*pdf417*) and `./zint -b84` (*micro_pdf417*), maybe the root of this bug is in *pdf417_enc()*. I tried to understand the problem, but there happens a lot and I didn't get it, sorry.

there is another bug in *pdf417.c* maybe this is related

```
./zint -b55 -i byteprocess-b55-b84

=================================================================
==176499==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffc
WRITE of size 4 at 0x7ffce769d850 thread T0
    #0 0x5556de91941b in byteprocess /zint-2.7.1/backend/pdf417.c:425
    #1 0x5556de91a475 in pdf417 /zint-2.7.1/backend/pdf417.c:645
    #2 0x5556de91fa78 in pdf417enc /zint-2.7.1/backend/pdf417.c:858
    #3 0x5556de8643ae in reduced_charset /zint-2.7.1/backend/library.c:853
    #4 0x5556de8643ae in extended_or_reduced_charset /zint-2.7.1/backend/l
    #5 0x5556de86a9b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #6 0x5556de86ed45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #7 0x5556de85e127 in main /zint-2.7.1/frontend/main.c:733
    #8 0x7f3882f3e022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #9 0x5556de86098d in _start (/zint-2.7.1/build-asan/frontend/zint+0x29

Address 0x7ffce769d850 is located in stack of thread T0 at offset 13296 in
    #0 0x5556de9194ff in pdf417 /zint-2.7.1/backend/pdf417.c:554

  This frame has 6 object(s):
    [48, 52) 'indexliste' (line 555)
    [64, 68) 'mclength' (line 556)
    [80, 220) 'dummy' (line 556)
    [288, 2368) 'mccorrection' (line 555)
    [2496, 13296) 'chainemc' (line 556) <== Memory access at offset 13296
    [13552, 14132) 'pattern' (line 557)
HINT: this may be a false positive if your program uses some custom stack
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /zint-2.7.1/backend/pdf41
Shadow bytes around the buggy address:
  0x10001cecbab0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbac0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbad0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbae0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbaf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x10001cecbb00: 00 00 00 00 00 00 00 00 00 00[f2]f2 f2 f2 f2 f2
  0x10001cecbb10: f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
  0x10001cecbb20: f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 00 00 00 00 00 00
  0x10001cecbb30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbb40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10001cecbb50: 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
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==176499==ABORTING
```

aztec.c -aztec_text_process()

```
/zint -b92 -i bin_append-b92

=================================================================
==176379==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffd
WRITE of size 1 at 0x7ffd1d40a540 thread T0
    #0 0x55d81c94f5d9 in bin_append_posn /zint-2.7.1/backend/common.c:70
    #1 0x55d81c94f6e6 in bin_append /zint-2.7.1/backend/common.c:57
    #2 0x55d81c8a0af1 in aztec_text_process /zint-2.7.1/backend/aztec.c:79
    #3 0x55d81c8a0af1 in aztec /zint-2.7.1/backend/aztec.c:1007
    #4 0x55d81c78aa79 in reduced_charset /zint-2.7.1/backend/library.c:859
    #5 0x55d81c78aa79 in extended_or_reduced_charset /zint-2.7.1/backend/l
    #6 0x55d81c7909b5 in ZBarcode_Encode /zint-2.7.1/backend/library.c:121
    #7 0x55d81c794d45 in ZBarcode_Encode_File /zint-2.7.1/backend/library.
    #8 0x55d81c784127 in main /zint-2.7.1/frontend/main.c:733
    #9 0x7f829b74c022 in __libc_start_main (/usr/lib/libc.so.6+0x27022)
    #10 0x55d81c78698d in _start (/zint-2.7.1/build-asan/frontend/zint+0x2

Address 0x7ffd1d40a540 is located in stack of thread T0 at offset 20176 in
    #0 0x55d81c89be1f in aztec /zint-2.7.1/backend/aztec.c:973

  This frame has 6 object(s):
    [48, 52) 'desc_data' (line 977)
    [64, 70) 'desc_ecc' (line 977)
    [96, 138) 'descriptor' (line 975)
    [176, 20176) 'binary_string' (line 975) <== Memory access at offset 20
    [20432, 40432) 'adjusted_string' (line 976)
    [40688, 60733) 'bit_pattern' (line 975)
HINT: this may be a false positive if your program uses some custom stack
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /zint-2.7.1/backend/commo
Shadow bytes around the buggy address:
  0x100023a79450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a79460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a79470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a79480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a79490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100023a794a0: 00 00 00 00 00 00 00 00[f2]f2 f2 f2 f2 f2 f2 f2
  0x100023a794b0: f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
  0x100023a794c0: f2 f2 f2 f2 f2 f2 f2 f2 00 00 00 00 00 00 00 00
  0x100023a794d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a794e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100023a794f0: 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
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==176379==ABORTING
```

If I understand it right *length* in *bin_append()* and *bin_append_posn()* is the length of the data you append and not the max length of the buffer
and the bug is in *aztec_text_process()* where you should guarantee that a call of *bin_append()* can't overflow your *binary_string*

I hope this helps you,
Nico

poc.tar.gz

SourceForge

Create a Project

Open Source Software

Business Software

Top Downloaded Projects

Company

About

Team

SourceForge Headquarters

225 Broadway Suite 1600

San Diego, CA 92101

+1 (858) 454-5900

Resources

Support

Site Documentation

Site Status