heap-buffer-overflow in function ttUSHORT() at stb_truetype.h:1286 #1286

Closed Vincebye opened this issue on Feb 16 · 1 comment

Vincebye commented on Feb 16

Describe
A heap-buffer-overflow was discovered in stb_truetype. The issue is being triggered in function ttUSHORT() at stb_truetype.h:1286
To Reproduce
test program

```
#include <stdio.h>
#include <stdlib.h>
#define STB_IMAGE_WRITE_IMPLEMENTATION
#include "stb_image_write.h"
#define STB_TRUETYPE_IMPLEMENTATION
#include "stb_truetype.h"
int main(int argc, const char *argv[])
{
    long int size = 0;
    unsigned char *fontBuffer = NULL;
    FILE *fontFile = fopen(argv[1], "rb");
    if (fontFile == NULL)
    {
        printf("Can not open font file!\n");
        return 0;
    }
    fseek(fontFile, 0, SEEK_END);
    size = ftell(fontFile);
    fseek(fontFile, 0, SEEK_SET);
    fontBuffer = calloc(size, sizeof(unsigned char));
    fread(fontBuffer, size, 1, fontFile);
    fclose(fontFile);
    stbtt_fontinfo info;
    if (!stbtt_InitFont(&info, fontBuffer, 0))
    {
        printf("stb init font failed\n");
    }
    int bitmap_w = 512;
    int bitmap_h = 128;
    free(fontBuffer);
    return 0;
}
```

Compile test program with address sanitizer with this command:

```
AFL_HARDEN=1 afl-gcc -I /src/stb/include ttf.c -o ttf -lm
```

You can get program here

Asan Reports

`./Asanttf crash/0`

Get ASan reports

```
==3156==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x63000000ebf0 at pc
0x55ba19faea04 bp 0x7ffc3b853000 sp 0x7ffc3b852ff0
READ of size 1 at 0x63000000ebf0 thread T0
    #0 0x55ba19faea03 in ttUSHORT /src/stb/include/stb_truetype.h:1286
    #1 0x55ba19faea03 in stbtt_InitFont_internal /src/stb/include/stb_truetype.h:1472
    #2 0x55ba19faea03 in stbtt_InitFont /src/stb/include/stb_truetype.h:4954
    #3 0x55ba19fb01fc in main /src/stb/ttf.c:32
    #4 0x7f443d2b80b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
    #5 0x55ba19f8970d in _start (/src/stb/Asanttf+0x470d)

0x63000000ebf0 is located 1 bytes to the right of 59375-byte region
[0x630000000400,0x63000000ebef)
allocated by thread T0 here:
    #0 0x7f443d686e17 in __interceptor_calloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:154
    #1 0x55ba19fb01c6 in main /src/stb/ttf.c:26
    #2 0x7f443d2b80b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

SUMMARY: AddressSanitizer: heap-buffer-overflow /src/stb/include/stb_truetype.h:1286 in ttUSHORT
Shadow bytes around the buggy address:
  0x0c607fff9d20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c607fff9d30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c607fff9d40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c607fff9d50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c607fff9d60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c607fff9d70: 00 00 00 00 00 00 00 00 00 00 00 00 00 07[fa]fa
  0x0c607fff9d80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c607fff9d90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c607fff9da0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c607fff9db0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c607fff9dc0: 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
  ==3156==ABORTING
```

Poc

Poc file is here

nothings commented on Feb 17 Owner

These are all well known:

stb/stb_truetype.h
Lines 4 to 11 in `af1a5bc`

```
// =======================================================================
//
//    NO SECURITY GUARANTEE -- DO NOT USE THIS ON UNTRUSTED FONT FILES
//
// This library does no range checking of the offsets found in the file,
// meaning an attacker can use it to read arbitrary memory.
//
// =======================================================================
```

The stb_truetype API does not know the length of the input font file and therefore cannot bounds check it.

nothings closed this as completed on Feb 17

This was referenced on Feb 17

heap-buffer-overflow in function stbtt__find_table at stb_truetype.h:1313 #1287

Closed

heap-buffer-overflow in function ttULONG() at stb_truetype.h:1288 #1288

Closed

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants