ChijinZ new cves ... on Jul 30, 2021 History

..

crashes	3 years ago
readme.md	last year

readme.md

Multiple vulnerabilities in tinyexr

There is a vulnerability in tinyexr (git repository: https://github.com/syoyo/tinyexr, Latest commit 65f9859 on Dec 31, 2018).

git log

```
commit 65f9859446c32f0dcfd91f376b1a5abafb8e0457
Author: Syoyo Fujita <syoyo@lighttransport.com>
Date:   Tue Jan 1 00:15:59 2019 +0900
```

CVE-2018-20652 Heap-buffer-overflow in function tinyexr::AllocateImage tinyexr.h:10302

I build tinyexr with clang and address sanitizer. When testcase (see: https://github.com/ChijinZ/security_advisories/blob/master/tinyexr_65f9859/crashes/heap-buffer-overflow-in-tinyexr.h:10302) is input into test_tinyexr (command: ./test_tinyexr testcase), a heap-buffer-overflow has triggered.

```
==27424==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000061f at pc 0x0000005bf456 bp 0x7ffe87d29050 sp
0x7ffe87d29048
WRITE of size 16 at 0x60200000061f thread T0
    #0 0x5bf455 in tinyexr::DecodePixelData(unsigned char**, int const*, unsigned char const*, unsigned long, int, int, int, int,
int, int, int, int, unsigned long, unsigned long, _EXRAttribute const*, unsigned long, _EXRChannelInfo const*, std::vector<unsigned
long, std::allocator<unsigned long> > const&) /home/jin/Documents/cve/tinyexr/./tinyexr.h:9837:22
    #1 0x584206 in tinyexr::DecodeTiledPixelData(unsigned char**, int*, int*, int const*, unsigned char const*, unsigned long, int,
int, int, int, int, int, int, int, unsigned long, unsigned long, _EXRAttribute const*, unsigned long, _EXRChannelInfo const*,
std::vector<unsigned long, std::allocator<unsigned long> > const&) /home/jin/Documents/cve/tinyexr/./tinyexr.h:10252:3
    #2 0x584206 in tinyexr::DecodeChunk(_EXRImage*, _EXRHeader const*, std::vector<unsigned long, std::allocator<unsigned long> >
const&, unsigned char const*, unsigned long, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >*)
/home/jin/Documents/cve/tinyexr/./tinyexr.h:10820
    #3 0x548e67 in tinyexr::DecodeEXRImage(_EXRImage*, _EXRHeader const*, unsigned char const*, unsigned char const*, unsigned
long, char const**) /home/jin/Documents/cve/tinyexr/./tinyexr.h:11091:15
    #4 0x548e67 in LoadEXRImageFromMemory /home/jin/Documents/cve/tinyexr/./tinyexr.h:11625
    #5 0x52f88e in LoadEXRImageFromFile /home/jin/Documents/cve/tinyexr/./tinyexr.h:11602:10
    #6 0x522f17 in LoadEXR /home/jin/Documents/cve/tinyexr/./tinyexr.h:11161:15
    #7 0x58ee40 in main /home/jin/Documents/cve/tinyexr/test_tinyexr.cc:130:13
    #8 0x7f45feacdb96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c:310
    #9 0x41bb29 in _start (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x41bb29)

0x60200000061f is located 11 bytes to the right of 4-byte region [0x602000000610,0x602000000614)
allocated by thread T0 here:
    #0 0x4c3973 in __interceptor_malloc (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x4c3973)
    #1 0x5a43fc in tinyexr::AllocateImage(int, _EXRChannelInfo const*, int const*, int, int)
/home/jin/Documents/cve/tinyexr/./tinyexr.h:10302:11

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/jin/Documents/cve/tinyexr/./tinyexr.h:9837:22 in
tinyexr::DecodePixelData(unsigned char**, int const*, unsigned char const*, unsigned long, int, int, int, int, int, int, int, int,
unsigned long, unsigned long, _EXRAttribute const*, unsigned long, _EXRChannelInfo const*, std::vector<unsigned long,
std::allocator<unsigned long> > const&)
Shadow bytes around the buggy address:
0x0c047fff8070: fa fa fd fd fa fa fd fd fa fa fd fa fa fa 01 fa
0x0c047fff8080: fa fa fd fa fa fa fd fa fa fa fd fa fa fa 01 fa
0x0c047fff8090: fa fa fd fd fa fa 00 00 fa fa fd fd fa fa 00 00
0x0c047fff80a0: fa fa fd fa fa fa 01 fa fa fa fd fd fa fa fd fa
0x0c047fff80b0: fa fa 01 fa fa fa fd fa fa fa fd fa fa fa 00 fa
=>0x0c047fff80c0: fa fa 04[fa]fa fa 04 fa fa fa 04 fa fa fa 04 fa
0x0c047fff80d0: fa fa 04 fa fa fa 04 fa fa fa 04 fa fa fa 04 fa
0x0c047fff80e0: fa fa 04 fa fa fa 04 fa fa fa 04 fa fa fa 04 fa
0x0c047fff80f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8100: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8110: 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
==27424==ABORTING
```

tinyexr::LoadEXRImageFromFile tinyexr.h:11593

I build tinyexr with clang and address sanitizer. When testcase (see: https://github.com/ChijinZ/security_advisories/blob/master/tinyexr_65f9859/crashes/heap-buffer-overflow-in-tinyexr.h:11593) is input into test_tinyexr (command: ./test_tinyexr testcase), a heap-buffer-overflow has triggered.

```
==28354==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x619000000e70 at pc 0x0000005bd8c1 bp 0x7ffd350c33d0 sp
0x7ffd350c33c8
READ of size 1 at 0x619000000e70 thread T0
    #0 0x5bd8c0 in tinyexr::cpy4(int*, int const*) /home/jin/Documents/cve/tinyexr/./tinyexr.h:7017:12
    #1 0x5bd8c0 in tinyexr::DecompressPiz(unsigned char*, unsigned char const*, unsigned long, unsigned long, int, _EXRChannelInfo
const*, int, int) /home/jin/Documents/cve/tinyexr/./tinyexr.h:9348
    #2 0x5bd8c0 in tinyexr::DecodePixelData(unsigned char**, int const*, unsigned char const*, unsigned long, int, int, int, int,
int, int, int, int, unsigned long, unsigned long, _EXRAttribute const*, unsigned long, _EXRChannelInfo const*, std::vector<unsigned
long, std::allocator<unsigned long> > const&) /home/jin/Documents/cve/tinyexr/./tinyexr.h:9641
    #3 0x585337 in tinyexr::DecodeChunk(_EXRImage*, _EXRHeader const*, std::vector<unsigned long, std::allocator<unsigned long> >
const&, unsigned char const*, unsigned long, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >*)
/home/jin/Documents/cve/tinyexr/./tinyexr.h:10902:20
    #4 0x548e67 in tinyexr::DecodeEXRImage(_EXRImage*, _EXRHeader const*, unsigned char const*, unsigned char const*, unsigned
long, char const**) /home/jin/Documents/cve/tinyexr/./tinyexr.h:11091:15
    #5 0x548e67 in LoadEXRImageFromMemory /home/jin/Documents/cve/tinyexr/./tinyexr.h:11625
    #6 0x52f88e in LoadEXRImageFromFile /home/jin/Documents/cve/tinyexr/./tinyexr.h:11602:10
    #7 0x522f17 in LoadEXR /home/jin/Documents/cve/tinyexr/./tinyexr.h:11161:15
    #8 0x58ee40 in main /home/jin/Documents/cve/tinyexr/test_tinyexr.cc:130:13
    #9 0x7f0adcba4b96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c:310
    #10 0x41bb29 in _start (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x41bb29)

0x619000000e70 is located 0 bytes to the right of 1008-byte region [0x619000000a80,0x619000000e70)
allocated by thread T0 here:
    #0 0x4f2bb2 in operator new(unsigned long) (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x4f2bb2)
    #1 0x52f844 in __gnu_cxx::new_allocator<unsigned char>::allocate(unsigned long, void const*) /usr/bin/../lib/gcc/x86_64-linux-
gnu/7.3.0/../../../../include/c++/7.3.0/ext/new_allocator.h:111:27
    #2 0x52f844 in std::allocator_traits<std::allocator<unsigned char> >::allocate(std::allocator<unsigned char>&, unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/alloc_traits.h:436
    #3 0x52f844 in std::_Vector_base<unsigned char, std::allocator<unsigned char> >::_M_allocate(unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:172
    #4 0x52f844 in std::_Vector_base<unsigned char, std::allocator<unsigned char> >::_M_create_storage(unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:187
    #5 0x52f844 in std::_Vector_base<unsigned char, std::allocator<unsigned char> >::_Vector_base(unsigned long,
std::allocator<unsigned char> const&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:138
    #6 0x52f844 in std::vector<unsigned char, std::allocator<unsigned char> >::vector(unsigned long, std::allocator<unsigned char>
const&) /usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:284
    #7 0x52f844 in LoadEXRImageFromFile /home/jin/Documents/cve/tinyexr/./tinyexr.h:11593
    #8 0x522f17 in LoadEXR /home/jin/Documents/cve/tinyexr/./tinyexr.h:11161:15
    #9 0x58ee40 in main /home/jin/Documents/cve/tinyexr/test_tinyexr.cc:130:13
    #10 0x7f0adcba4b96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c:310

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/jin/Documents/cve/tinyexr/./tinyexr.h:7017:12 in tinyexr::cpy4(int*, int
const*)
Shadow bytes around the buggy address:
0x0c327fff8170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c327fff8180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c327fff8190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c327fff81a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c327fff81b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c327fff81c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00[fa]fa
0x0c327fff81d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c327fff81e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c327fff81f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c327fff8200: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c327fff8210: 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
==28354==ABORTING
```

I build tinyexr with clang and address sanitizer. When testcase (see: https://github.com/ChijinZ/security_advisories/blob/master/tinyexr_65f9859/crashes/out-of-memory-in-tinyexr.h:11046) is input into test_tinyexr (command: ./test_tinyexr testcase), a out-of-memory has triggered.

```
==28640==ERROR: AddressSanitizer: allocator is out of memory trying to allocate 0x3f8000b80 bytes
    #0 0x4f2bb2 in operator new(unsigned long) (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x4f2bb2)
    #1 0x54833a in __gnu_cxx::new_allocator<unsigned long>::allocate(unsigned long, void const*) /usr/bin/../lib/gcc/x86_64-linux-
gnu/7.3.0/../../../../include/c++/7.3.0/ext/new_allocator.h:111:27
    #2 0x54833a in std::allocator_traits<std::allocator<unsigned long> >::allocate(std::allocator<unsigned long>&, unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/alloc_traits.h:436
    #3 0x54833a in std::_Vector_base<unsigned long, std::allocator<unsigned long> >::_M_allocate(unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:172
    #4 0x54833a in std::_Vector_base<unsigned long, std::allocator<unsigned long> >::_M_create_storage(unsigned long)
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:187
    #5 0x54833a in std::_Vector_base<unsigned long, std::allocator<unsigned long> >::_Vector_base(unsigned long,
std::allocator<unsigned long> const&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:138
    #6 0x54833a in std::vector<unsigned long, std::allocator<unsigned long> >::vector(unsigned long, std::allocator<unsigned long>
const&) /usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:284
    #7 0x54833a in tinyexr::DecodeEXRImage(_EXRImage*, _EXRHeader const*, unsigned char const*, unsigned char const*, unsigned
long, char const**) /home/jin/Documents/cve/tinyexr/./tinyexr.h:11046
    #8 0x54833a in LoadEXRImageFromMemory /home/jin/Documents/cve/tinyexr/./tinyexr.h:11625
    #9 0x52f88e in LoadEXRImageFromFile /home/jin/Documents/cve/tinyexr/./tinyexr.h:11602:10
    #10 0x522f17 in LoadEXR /home/jin/Documents/cve/tinyexr/./tinyexr.h:11161:15
    #11 0x58ee40 in main /home/jin/Documents/cve/tinyexr/test_tinyexr.cc:130:13
    #12 0x7f163a97fb96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c:310

==28640==HINT: if you don't care about these errors you may set allocator_may_return_null=1
SUMMARY: AddressSanitizer: out-of-memory (/home/jin/Documents/cve/tinyexr/test_tinyexr+0x4f2bb2) in operator new(unsigned long)
==28640==ABORTING
```

CVE-2020-18428 Out-of-range in function tinyexr::SaveEXR tinyexr.h:13107

I build tinyexr with clang and address sanitizer. When testcase (see: https://github.com/ChijinZ/security_advisories/blob/master/tinyexr_65f9859/crashes/out-of-range-in-tinyexr.h:13107) is input into test_tinyexr (command: ./test_tinyexr testcase), a out-of-range has triggered.

```
(gdb) bt
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
#1  0x00007ffff6aba801 in __GI_abort () at abort.c:79
#2  0x00007ffff7ad88b7 in ?? () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#3  0x00007ffff7adea06 in ?? () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#4  0x00007ffff7adea41 in std::terminate() () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#5  0x00007ffff7adec74 in __cxa_throw () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#6  0x00007ffff7ada7b5 in ?? () from /usr/lib/x86_64-linux-gnu/libstdc++.so.6
#7  0x000000000058df09 in std::vector<float, std::allocator<float> >::_M_range_check (this=<optimized out>, __n=0)
    at /usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:825
#8  std::vector<float, std::allocator<float> >::at (this=<optimized out>, __n=0) at /usr/bin/../lib/gcc/x86_64-linux-
gnu/7.3.0/../../../../include/c++/7.3.0/bits/stl_vector.h:846
#9  SaveEXR (data=<optimized out>, width=0, height=112, components=4, save_as_fp16=1, outfilename=0x5f38e0 <.str> "output.exr",
err=<optimized out>) at /home/jin/Documents/cve/tinyexr/./tinyexr.h:13107
#10 0x000000000058f01c in main (argc=<optimized out>, argv=<optimized out>) at test_tinyexr.cc:141
```