Closed Issue created 1 year ago by 4ugustus

Heap buffer overflow in cpContigBufToSeparateBuf, tiffcp.c:1373

Summary

There is a heap buffer overflow in cpContigBufToSeparateBuf in tools/tiffcp.c. Remote attackers could leverage this vulnerability to cause a denial-of-service via a crafted tiff file.

Version

LIBTIFF, Version 4.3.0, commit id fb61aee8

POC file

heap-buffer-overflow_tiffcp.c-1373

Steps to reproduce

```
# export CFLAGS="-O -g -fsanitize=address"
# export CXXFLAGS="-O -g -fsanitize=address"
# ./configure --prefix=$PWD/build_asan
# make -j64 & make install

# ./build_asan/bin/tiffcp -i -s -p separate poc /tmp/foo
TIFFReadDirectoryCheckOrder: Warning, Invalid TIFF directory; tags are not sorted in ascending order
poc: Warning, Nonstandard tile width 1, convert file.
TIFFFetchStripThing: Warning, Incorrect count for "StripOffsets"; tag ignored.
TIFFFetchStripThing: Warning, Incorrect count for "StripByteCounts"; tag ignored.
TIFFFillTile: 0: Invalid tile byte count, tile 1.
TIFFFillTile: 0: Invalid tile byte count, tile 2.
TIFFFillTile: 0: Invalid tile byte count, tile 3.
TIFFFillTile: 0: Invalid tile byte count, tile 4.
TIFFFillTile: 0: Invalid tile byte count, tile 5.
TIFFFillTile: 0: Invalid tile byte count, tile 6.
TIFFFillTile: 0: Invalid tile byte count, tile 7.
TIFFFillTile: 0: Invalid tile byte count, tile 8.
TIFFFillTile: 0: Invalid tile byte count, tile 9.
TIFFFillTile: 0: Invalid tile byte count, tile 10.
TIFFFillTile: 0: Invalid tile byte count, tile 11.
TIFFFillTile: 0: Invalid tile byte count, tile 12.
TIFFFillTile: 0: Invalid tile byte count, tile 13.
TIFFFillTile: 0: Invalid tile byte count, tile 14.
TIFFFillTile: 0: Invalid tile byte count, tile 15.
=================================================================
==44064==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61500000ff00 at pc 0x000000401e4
READ of size 1 at 0x61500000ff00 thread T0
    #0 0x401e4a in cpContigBufToSeparateBuf /root/programs/Libtiff/tools/tiffcp.c:1373
    #1 0x4034a2 in writeBufferToSeparateStrips /root/programs/Libtiff/tools/tiffcp.c:1683
    #2 0x404001 in cpImage /root/programs/Libtiff/tools/tiffcp.c:1420
    #3 0x4040c7 in cpContigTiles2SeparateStrips /root/programs/Libtiff/tools/tiffcp.c:1934
    #4 0x407600 in tiffcp /root/programs/Libtiff/tools/tiffcp.c:979
    #5 0x407600 in main /root/programs/Libtiff/tools/tiffcp.c:334
    #6 0x7fd77abcc83f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2083f)
    #7 0x401ab8 in _start (/root/programs/Libtiff/build_asan/bin/tiffcp+0x401ab8)

0x61500000ff00 is located 0 bytes to the right of 512-byte region [0x61500000fd00,0x61500000ff00)
allocated by thread T0 here:
    #0 0x7fd77b341602 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
    #1 0x7fd77b049c61 in _TIFFmalloc /root/programs/Libtiff/libtiff/tif_unix.c:314

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/programs/Libtiff/tools/tiffcp.c:1373 cpContigB
Shadow bytes around the buggy address:
  0x0c2a7fff9f90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff9fa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff9fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff9fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff9fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c2a7fff9fe0:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
```

```
  0x0c2a7fff9ff0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fffa000: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fffa010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fffa020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fffa030: 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
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  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
==44064==ABORTING
```

Platform

```
# uname -a
Linux 37d1a8efe7bb 4.15.0-142-generic #146~16.04.1-Ubuntu SMP Tue Apr 13 09:27:15 UTC 2021 x86_64 x8
```(Operating system, architecture, compiler details)`
```

Drag your designs here or click to upload.

**Tasks** 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

**Linked items** 0

Link issues together to show that they're related or that one is blocking others. Learn more.

**Related merge requests** 1

fix heap buffer overflow in tiffcp (#278)
!311

When this merge request is accepted, this issue will be closed automatically.

## Activity

4ugustus @waugustus · 8 months ago  Author  Contributor

I notice that the crash still exists in the latest version (5e180045, Fri Feb 25 10:38:31 2022 +0000). I have analyzed it to give a fix.

# Analysis

## Crash cause

This crash happens in tiffcp.c:1373
```

```
static void
cpContigBufToSeparateBuf(uint8_t* out, uint8_t* in,
                         uint32_t rows, uint32_t cols, int outskew, int inskew, tsample
_t spp,
                         int bytes_per_sample )
{
        while (rows-- > 0) {
                uint32_t j = cols;
                while (j-- > 0)
                {
                        int n = bytes_per_sample;

                        while( n-- ) {
                                *out++ = *in++;
                        }
                        in += (spp-1) * bytes_per_sample;
                }
                out += outskew;
                in += inskew;
        }
}
```

Obviously, this is **an out-of-bounds read error**. Since there is no check for the values of *rows*, *j*, and *n*, the pointers *in* and *out* can be moved back by any length.

From the code, we can see that the pointer *out* is moved back by *(bytes_per_sample * cols + outskew) * rows*, and *in* is moved back by *(bytes_per_sample * spp * cols + inskew) * rows* in this function.

In tiffcp.c:1683,

```
cpContigBufToSeparateBuf(obuf, (uint8_t*) buf + row * rowsize + s, nrows, imagewidth, 0, 0
```

we can get that,

```
bytes_per_sample = 1
cols = imagewidth
outskew = 0
inskew = 0
rows = nrows
```

So,

```
out = out + (bytes_per_sample * cols + outskew) * rows = out + imagewidth * nrows

in = in + (bytes_per_sample * spp * cols + inskew) * rows = in + spp * imagewidth * nrows
```

The buffer *in* and *out* are allocated in tiffcp.c:1416 and tiffcp.c:1671

```
tsize_t scanlinesize = TIFFRasterScanlineSize(in);
tsize_t bytes = scanlinesize * (tsize_t)imagelength;

buf = limitMalloc(bytes);
```

```
tsize_t stripsize = TIFFStripSize(out);

obuf = limitMalloc(stripsize);
```

Therefore, this crash happens when

```
if (imagewidth * nrows > stripsize || spp * imagewidth * nrows > bytes)
```

How to fix

We can add checks for such out-of-bounds read error.

4ugustus mentioned in merge request !311 (merged) 8 months ago

p870613 mentioned in issue #397 (closed) 8 months ago

4ugustus mentioned in commit 88d79a45 8 months ago

Even Rouault closed via commit 408976c4 8 months ago