Bug 2831 - tiffcrop.c:9206:heap buffer overflow in invertImage

Status: RESOLVED FIXED
Reported: 2018-12-17 00:57 by jontsang
Modified: 2019-02-19 09:37 (History)

Product: libtiff
Component: default
Version: unspecified
Platform: PC Linux

Importance: P1 critical
Target Milestone: ---
Assigned To: Frank Warmerdam

URL:
Whiteboard:
Keywords:

Depends on:
Blocks:
Show dependency tree / graph

Attachments	
poc_invertImage (174 bytes, image/tiff) 2018-12-17 21:09, jontsang	Details
Add an attachment (proposed patch, testcase, etc.)	

Note

You need to log in before you can comment on or make changes to this bug.

Description From jontsang 2018-12-17 00:57:32

```
on libtiff 4.0.10 (the latest version):

The invertImage()  function in tiffcrop.c:9206 allows remote
attackers to cause a denial of service (heap buffer overflow) via invert color
space.

~/src/tiff-4.0.10/build/bin/tiffcrop -I data poc.tiff out.tiff
TIFFReadDirectory: Warning, Bogus "StripByteCounts" field, ignoring and
calculating from imagelength.
=================================================================
==101572==ERROR: AddressSanitizer: heap-buffer-overflow on address
0x60b00000aff7 at pc 0x00000042baf7 bp 0x7ffe523bd140 sp 0x7ffe523bd130
READ of size 1 at 0x60b00000aff7 thread T0
    #0 0x42baf6 in invertImage /root/src/tiff-4.0.10/tools/tiffcrop.c:9206
    #1 0x4263a8 in createCroppedImage
/root/src/tiff-4.0.10/tools/tiffcrop.c:7666
    #2 0x40afc3 in main /root/src/tiff-4.0.10/tools/tiffcrop.c:2378
    #3 0x7f2c2623a82f in __libc_start_main
(/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #4 0x4033f8 in _start
(/root/src/tiff-4.0.10/build-asan/bin/tiffcrop+0x4033f8)

0x60b00000aff7 is located 0 bytes to the right of 103-byte region
[0x60b00000af90,0x60b00000aff7)
allocated by thread T0 here:
    #0 0x7f2c27484602 in malloc
(/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
    #1 0x48aa7b in _TIFFmalloc /root/src/tiff-4.0.10/libtiff/tif_unix.c:314
    #2 0x41f68f in loadImage /root/src/tiff-4.0.10/tools/tiffcrop.c:6138
    #3 0x40ae9d in main /root/src/tiff-4.0.10/tools/tiffcrop.c:2348
    #4 0x7f2c2623a82f in __libc_start_main
(/lib/x86_64-linux-gnu/libc.so.6+0x2082f)

SUMMARY: AddressSanitizer: heap-buffer-overflow
/root/src/tiff-4.0.10/tools/tiffcrop.c:9206 invertImage
Shadow bytes around the buggy address:
  0x0c167fff95a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff95b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff95c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff95d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff95e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c167fff95f0: fa fa 00 00 00 00 00 00 00 00 00 00 00 00[07]fa
  0x0c167fff9600: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff9610: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff9620: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff9630: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c167fff9640: 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
==101572==ABORTING

=========================
Credit:  Alpha Lab of Topsec
```

------- *Comment #1 From jontsang 2018-12-17 21:09:13* -------

Created an attachment (id=882) [details]
poc_invertImage

------- *Comment #2 From Thomas Bernard 2019-02-11 17:01:21* -------

OK. The code in invertImage() has several issues :

```
    case 2: for (row = 0; row < length; row++)
              for (col = 0; col < width; col++)
                {
        bytebuff1 = 4 - (uint8)(*src & 192 >> 6);
        bytebuff2 = 4 - (uint8)(*src & 48  >> 4);
        bytebuff3 = 4 - (uint8)(*src & 12  >> 2);
        bytebuff4 = 4 - (uint8)(*src & 3);
```

```
        *src = (bytebuff1 << 6) | (bytebuff2 << 4) | (bytebuff3 << 2) |
bytebuff4;
                src++;
                }
            break;
```

https://gitlab.com/libtiff/libtiff/blob/master/tools/tiffcrop.c#L9205

```
1) first of all the values are false.
it should be (3 - x) not (4 - x).
A 0 byte (4 pixels to zero) will be inverted to binary 01010100, not 11111111 !
For some unknown reason the same bug is for bps values 2 and 4, but not for bps
values 32, 16, 8 and 1

2) a loop iteration is processing 4 pixels, but width iterations are executed,
so 4*width pixels are inverted, thus the heap buffer overflow.
it should be
for (col = 0; col < width; col += 8/bps)
```

------- Comment #3 From Thomas Bernard 2019-02-11 17:13:49 -------

See merge request
https://gitlab.com/libtiff/libtiff/merge_requests/61

------- Comment #4 From Even Rouault 2019-02-19 09:37:23 -------

Fixed per https://gitlab.com/libtiff/libtiff/merge_requests/61
