Closed Issue created 10 months ago by 4ugustus

tiffset: Global-buffer-overflow in _TIFFmemcpy, tif_unix.c:346

Summary

There is a global buffer overflow in _TIFFmemcpy in libtiff/tif_unix.c:346. 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 cd57b554 (Wed Dec 29 18:43:34 2021 +0000)

Steps to reproduce

```
# ./build_asan/bin/tiffset -s 93 helloworld ./poc
TIFFReadDirectoryCheckOrder: Warning, Invalid TIFF directory; tags are not sorted in ascending order
TIFFReadDirectory: Warning, Unknown field with tag 2 (0x2) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 0 (0x0) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 65280 (0xff00) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 250 (0xfa) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 9 (0x9) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 15 (0xf) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 65535 (0xffff) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 23901 (0x5d5d) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 93 (0x5d) encountered.
TIFFReadDirectory: Warning, Unknown field with tag 58624 (0xe500) encountered.
TIFFReadDirectory: Warning, Ignoring TransferFunction since BitsPerSample tag not found.
TIFFReadDirectory: Warning, Invalid data type for tag StripByteCounts.
TIFFReadDirectory: Warning, Invalid data type for tag TileOffsets.
TIFFFetchStripThing: Warning, Incorrect count for "TileOffsets"; tag ignored.
TIFFReadDirectory: Warning, Wrong "StripByteCounts" field, ignoring and calculating from imagelength
=================================================================
==62478==ERROR: AddressSanitizer: global-buffer-overflow on address 0x0000004030e7 at pc 0x7f851f5cb
READ of size 2053925041 at 0x0000004030e7 thread T0
    #0 0x7f851f5cb934 in __asan_memcpy (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x8c934)
    #1 0x7f851f2dd249 in _TIFFmemcpy /root/test4/libtiff/libtiff/tif_unix.c:346
    #2 0x7f851f1e398c in setByteArray /root/test4/libtiff/libtiff/tif_dir.c:54
    #3 0x7f851f1eaa9f in _TIFFVSetField /root/test4/libtiff/libtiff/tif_dir.c:592
    #4 0x7f851f1ed75d in TIFFVSetField /root/test4/libtiff/libtiff/tif_dir.c:890
    #5 0x7f851f1ed18d in TIFFSetField /root/test4/libtiff/libtiff/tif_dir.c:834
    #6 0x401ab0 in main /root/test4/libtiff/tools/tiffset.c:149
    #7 0x7f851ee0683f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2083f)
    #8 0x4012e8 in _start (/root/test4/libtiff/build_asan/bin/tiffset+0x4012e8)

0x0000004030e7 is located 57 bytes to the left of global variable '*.LC0' defined in 'tiffset.c' (0x
  '*.LC0' is ascii string '%s

'
0x0000004030e7 is located 0 bytes to the right of global variable 'usageMsg' defined in 'tiffset.c:4
  'usageMsg' is ascii string 'Set the value of a TIFF header to a specified value

usage: tiffset [options] filename
where options are:
 -s  <tagname> [count] <value>...   set the tag value
 -u  <tagname> to unset the tag
 -d  <dirno> set the directory
 -sd <diroff> set the subdirectory
 -sf <tagname> <filename>  read the tag value from file (for ASCII tags only)
 -h  this help screen
'
SUMMARY: AddressSanitizer: global-buffer-overflow ??:0 __asan_memcpy
Shadow bytes around the buggy address:
  0x0000800785c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800785d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800785e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800785f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x000080078600: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x000080078610: 00 00 00 00 00 00 00 00 00 00 00 00[07]f9 f9 f9
```

```
  0x000080078620: f9 f9 f9 f9 05 f9 f9 f9 f9 f9 f9 f9 00 00 00 00
  0x000080078630: 04 f9 f9 f9 f9 f9 f9 f9 03 f9 f9 f9 f9 f9 f9 f9
  0x000080078640: 03 f9 f9 f9 f9 f9 f9 f9 00 00 00 04 f9 f9 f9 f9
  0x000080078650: 04 f9 f9 f9 f9 f9 f9 f9 00 00 00 00 f9 f9 f9 f9
  0x000080078660: 03 f9 f9 f9 f9 f9 f9 f9 00 00 04 f9 f9 f9 f9 f9
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
==62478==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
```

POC File

tiffset_poc

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 the global-buffer-overflow in tiffset

!287

Activity

4ugustus @waugustus · 10 months ago

Author Contributor

Hi,

I have analyzed the cause of this crash and found that it is interesting.

Root cause

The code that triggers the vulnerability is in _TIFFmemcpy, tif_unix.c:346:

```
memcpy(d, s, (size_t) c).
```

From GDB, we can find that the value of `c` is tremendous which is weird.

```
gdb-peda$ p c
$10 = 0xffffe899
```

By tracking the assignment process of `c`, we can locate the problem to _TIFFVSetField, tif_dir.c:576-593.

```
if (fip->field_type == TIFF_ASCII)
{
    uint32_t ma;
    char* mb;
    if (fip->field_passcount)
    {
        assert(fip->field_writecount==TIFF_VARIABLE2);
        ma=(uint32_t)va_arg(ap, uint32_t);
        mb=(char*)va_arg(ap,char*);
    }
    else
    {
        mb=(char*)va_arg(ap,char*);
        ma=(uint32_t)(strlen(mb) + 1);
    }
    tv->setByteArray(&tv->value,mb,ma,1); // crash!
    count=ma;
}
```

Since c is equal to `ma` times `1` (setByteArray, tif_dir.c:50), the reason for the overflow is that `ma` is incorrectly assigned as follows.

`ma=(uint32_t)va_arg(ap, uint32_t);`

From GDB, we can print the variable parameters pointed by `ap`

```
gdb-peda$ p ap.reg_save_area + ap.gp_offset
$18 = (void *) 0x7fffffffe470

gdb-peda$ x/32xb 0x7fffffffe470
0x7fffffffe470: 0x99    0xe8    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffe478: 0x04    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffe480: 0x05    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffe488: 0x5d    0x00    0x00    0x00    0x00    0x00    0x00    0x00
```

Note that 0x7fffffffe899 is the address of argv[3] ("helloworld"). Since `va_arg` reads a variable of type `uint32_t`, it returns the last four bytes of the address (i.e., **0xffffe899**).

So the **root cause** for the crash is that the dir count is not passed when `TIFFSetField` is called (main, tiffset.c:149), which causes `va_arg` to mistakenly read the address of the string ("helloworld") as the dir count (_TIFFVSetField, tif_dir.c:583).

Triggering conditions

Two conditions need to be met to trigger this crash:

1. The tag specified by `-s` must be the **custom** tag, and it must exist in the tiff file. (`field_passcount = True`)
2. The second word of the DE field must be `0x0200`. (`field_type = TIFF_ASCII`)

How to fix

TIFFFieldPassCount(fip) can be used to determine whether `count` needs to be passed.

```
if (TIFFFieldDataType(fip) == TIFF_ASCII) {
    if(TIFFFieldPassCount( fip )) { // need to pass count
        size_t len;
        len = (uint32_t)(strlen(argv[arg_index] + 1));
        if (TIFFSetField(tiff, TIFFFieldTag(fip), (uint16_t)len, argv[arg_index]) != 1)
            fprintf( stderr, "Failed to set %s=%s",
```

```
                TIFFFieldName(fip), argv[arg_index] );
    } else { // not need to pass count
        if (TIFFSetField(tiff, TIFFFieldTag(fip), argv[arg_index]) != 1)
            fprintf( stderr, "Failed to set %s=%s",
                TIFFFieldName(fip), argv[arg_index] );
    }
}
```

Edited by 4ugustus 10 months ago

4ugustus mentioned in merge request !287 (merged) 10 months ago

Timothy Lyanguzov @theta682 · 10 months ago Contributor

CVE-2022-22844 is assigned to this issue.

Timothy Lyanguzov @theta682 · 10 months ago Contributor

@bobfriesenhahn the fix require a new release.

4ugustus closed via commit `03047a26` 10 months ago

4ugustus mentioned in commit `freedesktop-sdk/mirrors/gitlab/libtiff/libtiff@03047a26` 10 months ago

Ross Burton @rossburton · 10 months ago

Is it expected that there will be a new release shortly?

Timothy Lyanguzov @theta682 · 9 months ago Contributor

When the new release with the fix is expected?

Timothy Lyanguzov @theta682 · 9 months ago Contributor

See CVE-2022-22844. This fix needs a new release. @bobfriesenhahn can you prepare a 4.3.1 release?

Please register or sign in to reply