heap-buffer-overflow in function jfif_decode at jfif.c:546 #24

Open xiaoxiongwang opened this issue on May 23, 2020 · 2 comments

xiaoxiongwang commented on May 23, 2020

Tested in Ubuntu 16.04, 64bit.

The tesecase is heap-buffer-overflow_ffjpeg_d1.

I use the following command:

```
ffjpeg -d heap-buffer-overflow_ffjpeg_d1
```

and get:

```
Segmentation fault
```

I use **valgrind** to analysis the bug and get the below information (absolute path information omitted):

```
==22952== Memcheck, a memory error detector
==22952== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==22952== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==22952== Command: ffjpeg -d heap-buffer-overflow_ffjpeg_d1
==22952==
==22952== Conditional jump or move depends on uninitialised value(s)
==22952==    at 0x40E6D0: yuv_to_rgb (color.c:26)
==22952==    by 0x40BB0F: jfif_decode (jfif.c:546)
==22952==    by 0x400E3A: main (ffjpeg.c:24)
==22952==
==22952== Conditional jump or move depends on uninitialised value(s)
==22952==    at 0x40E759: yuv_to_rgb (color.c:27)
==22952==    by 0x40BB0F: jfif_decode (jfif.c:546)
==22952==    by 0x400E3A: main (ffjpeg.c:24)
==22952==
==22952== Conditional jump or move depends on uninitialised value(s)
==22952==    at 0x40E646: yuv_to_rgb (color.c:25)
==22952==    by 0x40BB0F: jfif_decode (jfif.c:546)
==22952==    by 0x400E3A: main (ffjpeg.c:24)
==22952==
==22952== Invalid read of size 4
==22952==    at 0x40BB00: jfif_decode (jfif.c:546)
==22952==    by 0x400E3A: main (ffjpeg.c:24)
==22952==  Address 0x521f058 is 0 bytes after a block of size 21,384 alloc'd
==22952==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==22952==    by 0x409DB9: jfif_decode (jfif.c:443)
==22952==    by 0x400E3A: main (ffjpeg.c:24)
==22952==
==22952== Syscall param write(buf) points to uninitialised byte(s)
==22952==    at 0x4F312C0: __write_nocancel (syscall-template.S:84)
==22952==    by 0x4EB2BFE: _IO_file_write@@GLIBC_2.2.5 (fileops.c:1263)
==22952==    by 0x4EB4408: new_do_write (fileops.c:518)
==22952==    by 0x4EB4408: _IO_do_write@@GLIBC_2.2.5 (fileops.c:494)
==22952==    by 0x4EB347C: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1331)
==22952==    by 0x4EA87BA: fwrite (iofwrite.c:39)
==22952==    by 0x401AE2: bmp_save (bmp.c:97)
==22952==    by 0x400E4F: main (ffjpeg.c:26)
==22952==  Address 0x52dffe8 is 56 bytes inside a block of size 4,096 alloc'd
==22952==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==22952==    by 0x4EA71D4: _IO_file_doallocate (filedoalloc.c:127)
==22952==    by 0x4EB5593: _IO_doallocbuf (genops.c:398)
==22952==    by 0x4EB48F7: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:820)
==22952==    by 0x4EB328C: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1331)
==22952==    by 0x4EA87BA: fwrite (iofwrite.c:39)
==22952==    by 0x401A30: bmp_save (bmp.c:93)
==22952==    by 0x400E4F: main (ffjpeg.c:26)
==22952==
==22952== Syscall param write(buf) points to uninitialised byte(s)
==22952==    at 0x4F312C0: __write_nocancel (syscall-template.S:84)
==22952==    by 0x4EB2BFE: _IO_file_write@@GLIBC_2.2.5 (fileops.c:1263)
==22952==    by 0x4EB4408: new_do_write (fileops.c:518)
==22952==    by 0x4EB4408: _IO_do_write@@GLIBC_2.2.5 (fileops.c:494)
==22952==    by 0x4EB39AF: _IO_file_close_it@@GLIBC_2.2.5 (fileops.c:165)
==22952==    by 0x4EA73EE: fclose@@GLIBC_2.2.5 (iofclose.c:58)
==22952==    by 0x401B63: bmp_save (bmp.c:99)
==22952==    by 0x400E4F: main (ffjpeg.c:26)
==22952==  Address 0x52dffb0 is 0 bytes inside a block of size 4,096 alloc'd
==22952==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==22952==    by 0x4EA71D4: _IO_file_doallocate (filedoalloc.c:127)
==22952==    by 0x4EB5593: _IO_doallocbuf (genops.c:398)
==22952==    by 0x4EB48F7: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:820)
==22952==    by 0x4EB328C: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1331)
==22952==    by 0x4EA87BA: fwrite (iofwrite.c:39)
==22952==    by 0x401A30: bmp_save (bmp.c:93)
==22952==    by 0x400E4F: main (ffjpeg.c:26)
==22952==
==22952==
==22952== HEAP SUMMARY:
==22952==     in use at exit: 0 bytes in 0 blocks
==22952==   total heap usage: 19 allocs, 19 frees, 9,423,684 bytes allocated
==22952==
==22952== All heap blocks were freed -- no leaks are possible
==22952==
==22952== For counts of detected and suppressed errors, rerun with: -v
==22952== Use --track-origins=yes to see where uninitialised values come from
```

```
==22952== ERROR SUMMARY: 776684 errors from 6 contexts (suppressed: 0 from 0)
```

I use **AddressSanitizer** to build ffjpeg and running it with the following command:

```
ffjpeg -e heap-buffer-overflow_ffjpeg_d1
```

This is the ASAN information (absolute path information omitted):

```
=================================================================
==687==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000efd0 at pc 0x000000405c60 bp 0x7ffccf9b8f90 sp 0x7ffccf9b8f80
READ of size 4 at 0x60200000efd0 thread T0
    #0 0x405c5f in jfif_decode ffjpeg/src/jfif.c:546
    #1 0x401233 in main (ffjpeg/src/ffjpeg+0x401233)
    #2 0x7f0380f4582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #3 0x4010c8 in _start (ffjpeg/src/ffjpeg+0x4010c8)

0x60200000efd1 is located 0 bytes to the right of 1-byte region [0x60200000efd0,0x60200000efd1)
allocated by thread T0 here:
    #0 0x7f0381387662 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98662)
    #1 0x404c01 in jfif_decode ffjpeg/src/jfif.c:444
    #2 0x401233 in main (ffjpeg/src/ffjpeg+0x401233)
    #3 0x7f0380f4582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)

SUMMARY: AddressSanitizer: heap-buffer-overflow ffjpeg/src/jfif.c:546 jfif_decode
Shadow bytes around the buggy address:
  0x0c047fff9da0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9db0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9dc0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9dd0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9de0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c047fff9df0: fa fa fa fa fa fa fa fa fa fa[01]fa fa fa 01 fa
  0x0c047fff9e00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9e10: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9e20: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9e30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9e40: 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
==687==ABORTING
```

The gdb reports (absolute path information omitted)::

```
Starting program: ffjpeg -d heap-buffer-overflow_ffjpeg_d1

Program received signal SIGSEGV, Segmentation fault.

[----------------------------------registers-----------------------------------]
RAX: 0x82f4
RBX: 0x7ffff77cd47c --> 0x0
RCX: 0x7ffff7f5b420 --> 0x0
RDX: 0x215c5400 ('')
RSI: 0xe
RDI: 0x0
RBP: 0x8480
RSP: 0x7fffffffd570 --> 0x0
RIP: 0x40bb00 (<jfif_decode+11520>:     mov    esi,DWORD PTR [r9+rax*4])
R8 : 0x7ffff7f5b41f --> 0x0
R9 : 0x622430 --> 0x0
R10: 0xff
R11: 0x215c5400 ('')
R12: 0x7ffff7f5b41e --> 0x0
R13: 0x622010 --> 0x202000000020 (' ')
R14: 0x1b
R15: 0xe8a
EFLAGS: 0x10212 (carry parity ADJUST zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x40baf6 <jfif_decode+11510>:        mov    edx,r11d
   0x40baf9 <jfif_decode+11513>:        cdqe
   0x40bafb <jfif_decode+11515>:        add    rax,QWORD PTR [rsp+0x10]
=> 0x40bb00 <jfif_decode+11520>:        mov    esi,DWORD PTR [r9+rax*4]
   0x40bb04 <jfif_decode+11524>:        mov    r9,r12
   0x40bb07 <jfif_decode+11527>:        add    r12,0x3
   0x40bb0b <jfif_decode+11531>:        call   0x40e5c0 <yuv_to_rgb>
   0x40bb10 <jfif_decode+11536>:        mov    ecx,DWORD PTR [r13+0x0]
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffd570 --> 0x0
0008| 0x7fffffffd578 --> 0x200
0016| 0x7fffffffd580 --> 0x82da
0024| 0x7fffffffd588 --> 0x6221d0 --> 0xe0000000e
0032| 0x7fffffffd590 --> 0xe00000000
0040| 0x7fffffffd598 --> 0x680000000e
0048| 0x7fffffffd5a0 --> 0x7ffff71eb010 --> 0xe8db8effba253b
0056| 0x7fffffffd5a8 --> 0x4a00000080
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x000000000040bb00 in jfif_decode (ctxt=ctxt@entry=0x622010, pb=pb@entry=0x7fffffffd840) at jfif.c:546
546                 yuv_to_rgb(*ysrc, *vsrc, *usrc, bdst + 2, bdst + 1, bdst + 0);
gdb-peda$ bt
```

```
#0  0x000000000040bb00 in jfif_decode (ctxt=ctxt@entry=0x622010, pb=pb@entry=0x7fffffffd840) at jfif.c:546
#1  0x0000000000400e3b in main (argc=argc@entry=0x3, argv=argv@entry=0x7fffffffd948) at ffjpeg.c:24
#2  0x00007ffff7a2d830 in __libc_start_main (main=0x400be0 <main>, argc=0x3, argv=0x7fffffffd948, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>,
stack_end=0x7fffffffd938)
    at ../csu/libc-start.c:291
#3  0x0000000000401019 in _start ()
```

An attacker can exploit this vulnerability by submitting a malicious bmp that exploits this bug which will result in a Denial of Service (DoS).

xiaoxiongwang commented on May 29, 2020 Author

CVE-2020-13439 has been assigned to this issue.The link is here.

rockcarry commented on Jul 27, 2020 Owner

lastest code can't reprodeuce the issue.
last commit: `31649ad`
@xiaoxiongwang please check and test.

rockcarry added a commit that referenced this issue on Aug 3, 2020

`fix issue #24.` `3dddf98`

Marsman1996 mentioned this issue on Dec 1, 2021

Heap-buffer-overflows in jfif_decode() at jfif.c:552:31 and 552:38 #43

Closed

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

2 participants