SEGV in function jfif_encode in jfif.c:748 #23

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

xiaoxiongwang commented on May 23, 2020 • edited

Tested in Ubuntu 16.04, 64bit.

The testcase is segv_ffjpeg_e2.

I use the following command:

```
ffjpeg -e segv_ffjpeg_e2
```

and get:

```
Segmentation fault
```

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

```
==12529== Memcheck, a memory error detector
==12529== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==12529== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==12529== Command: ffjpeg -e segv_ffjpeg_e2
==12529==
==12529== Argument 'size' of function malloc has a fishy (possibly negative) value: -2097127520
==12529==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==12529==    by 0x4015EB: bmp_load (bmp.c:52)
==12529==    by 0x400F2B: main (ffjpeg.c:29)
==12529==
==12529== Invalid read of size 1
==12529==    at 0x40C930: jfif_encode (jfif.c:748)
==12529==    by 0x400F33: main (ffjpeg.c:30)
==12529==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==12529==
==12529==
==12529== Process terminating with default action of signal 11 (SIGSEGV)
==12529==  Access not within mapped region at address 0x0
==12529==    at 0x40C930: jfif_encode (jfif.c:748)
==12529==    by 0x400F33: main (ffjpeg.c:30)
==12529==  If you believe this happened as a result of a stack
==12529==  overflow in your program's main thread (unlikely but
==12529==  possible), you can try to increase the size of the
==12529==  main thread stack using the --main-stacksize= flag.
==12529==  The main thread stack size used in this run was 8388608.
==12529==
==12529== HEAP SUMMARY:
==12529==     in use at exit: 33,643,096 bytes in 12 blocks
==12529==   total heap usage: 14 allocs, 2 frees, 33,647,744 bytes allocated
==12529==
==12529== LEAK SUMMARY:
==12529==    definitely lost: 0 bytes in 0 blocks
==12529==    indirectly lost: 0 bytes in 0 blocks
==12529==      possibly lost: 0 bytes in 0 blocks
==12529==    still reachable: 33,643,096 bytes in 12 blocks
==12529==         suppressed: 0 bytes in 0 blocks
==12529== Rerun with --leak-check=full to see details of leaked memory
==12529==
==12529== For counts of detected and suppressed errors, rerun with: -v
==12529== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
Segmentation fault
```

The gdb reports:

```
Starting program: ffjpeg -e segv_ffjpeg_e2

Program received signal SIGSEGV, Segmentation fault.

[----------------------------------registers-----------------------------------]
RAX: 0x3f800010
RBX: 0x7ffff000e3e0 --> 0x0
RCX: 0xff
RDX: 0x7f000020
RSI: 0x3f800010
RDI: 0xfe000040
RBP: 0x7ffff00103f0 --> 0x0
RSP: 0x7fffffffd6e0 --> 0x7fffffffd8c0 --> 0xff7f000020
RIP: 0x40c930 (<jfif_encode+1968>:      movzx  edx,BYTE PTR [r14])
R8 : 0x1fc000080
R9 : 0x7ffff00063d0 --> 0x0
R10: 0x7fffffffd8c0 --> 0xff7f000020
R11: 0x7ffff0004ecc --> 0x105010000000000
R12: 0x7ffff00063d0 --> 0x0
R13: 0x7fffffffd770 --> 0x0
R14: 0x0
R15: 0x0
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x40c91c <jfif_encode+1948>: mov    rdx,QWORD PTR [rsp]
   0x40c920 <jfif_encode+1952>: lea    rsp,[rsp+0x98]
   0x40c928 <jfif_encode+1960>: nop    DWORD PTR [rax+rax*1+0x0]
=> 0x40c930 <jfif_encode+1968>: movzx  edx,BYTE PTR [r14]
   0x40c934 <jfif_encode+1972>: movzx  esi,BYTE PTR [r14+0x1]
   0x40c939 <jfif_encode+1977>: mov    rcx,r12
   0x40c93c <jfif_encode+1980>: movzx  edi,BYTE PTR [r14+0x2]
   0x40c941 <jfif_encode+1985>: mov    r9,rbp
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffd6e0 --> 0x7fffffffd8c0 --> 0xff7f000020
0008| 0x7fffffffd6e8 --> 0x7fff00000000
0016| 0x7fffffffd6f0 --> 0x1fc000080
0024| 0x7fffffffd6f8 --> 0xfe000040
0032| 0x7fffffffd700 --> 0x7fff00000100
0040| 0x7fffffffd708 --> 0x7f000020
0048| 0x7fffffffd710 --> 0x7ffff00008c0 --> 0xff7f000020
0056| 0x7fffffffd718 --> 0xff000000ff00
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
jfif_encode (pb=pb@entry=0x7fffffffd8c0) at jfif.c:748
748                 rgb_to_yuv(bsrc[2], bsrc[1], bsrc[0], ydst, udst, vdst);
gdb-peda$ bt
#0  jfif_encode (pb=pb@entry=0x7fffffffd8c0) at jfif.c:748
#1  0x0000000000400f34 in main (argc=argc@entry=0x3, argv=argv@entry=0x7fffffffd9c8)
    at ffjpeg.c:30
#2  0x00007ffff7a2d830 in __libc_start_main (main=0x400be0 <main>, argc=0x3,
    argv=0x7fffffffd9c8, init=<optimized out>, fini=<optimized out>,
    rtld_fini=<optimized out>, stack_end=0x7fffffffd9b8) 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 • edited — Author

CVE-2020-13438 has been assigned to this issue.The link is CVE-2020-13438.

rockcarry added a commit that referenced this issue on Jul 27, 2020

`fix issue #23.` 31649ad

rockcarry commented on Jul 27, 2020 — Owner

new commit `31649ad` fix this issue.
@xiaoxiongwang please check and test.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

2 participants