brackeen / ok-file-formats Public

Code Issues Pull requests Actions Projects Security Insights

New issue

Jump to bottom

heap-buffer-overflow in function ok_jpg_decode_block_progressive() at ok_jpg.c:1054 #8

Closed WayneDevMaze opened this issue on Jun 26, 2020 · 1 comment

WayneDevMaze commented on Jun 26, 2020

Describe

A heap-buffer-overflow was discovered in ok_file_formats. The issue is being triggered in function ok_jpg_decode_block_progressive() at ok_jpg.c:1054

Reproduce

test program

```
#include <stdio.h>
#include <stdlib.h>
#include "ok_mo.h"
#include "ok_jpg.h"
int main(int _argc, char **_argv) {
    FILE *file = fopen(_argv[1], "rb");
    ok_jpg image = ok_jpg_read(file, OK_JPG_COLOR_FORMAT_RGBA);
    fclose(file);
    if (image.data) {
        printf("Got image! Size: %li x %li\n", (long)image.width, (long)image.height);
        free(image.data);
    }
    return 0;
}
```

Tested in Ubuntu 18.04, 64bit.
Compile test program with address sanitizer with this command:

```
gcc -g -fsanitize=address -fno-omit-frame-pointer -O1 -o Asanjpg main.c ok_jpg.c  ok_jpg.h
```

You can get program here.

ASan Reports

```
./Asanjpg crash/jpg-heap-buffer-overflow-2
```

Get ASan reports

```
==78746==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x631000011618 at pc 0x556e3ec8f64c bp 0x7ffca25f9440 sp 0x7ffca25f9430
WRITE of size 2 at 0x631000011618 thread T0
    #0 0x556e3ec8f64b in ok_jpg_decode_block_progressive /root/study/ok-file-formats/afl-test/ok_jpg.c:1054
    #1 0x556e3ec90c16 in ok_jpg_decode_scan /root/study/ok-file-formats/afl-test/ok_jpg.c:1217
    #2 0x556e3ec95c60 in ok_jpg_read_sos /root/study/ok-file-formats/afl-test/ok_jpg.c:1734
    #3 0x556e3ec96d3c in ok_jpg_decode2 /root/study/ok-file-formats/afl-test/ok_jpg.c:1900
    #4 0x556e3ec97605 in ok_jpg_decode /root/study/ok-file-formats/afl-test/ok_jpg.c:1990
    #5 0x556e3ec868a4 in ok_jpg_read_with_allocator /root/study/ok-file-formats/afl-test/ok_jpg.c:268
    #6 0x556e3ec8671b in ok_jpg_read /root/study/ok-file-formats/afl-test/ok_jpg.c:257
    #7 0x556e3ec85d5e in main /root/study/ok-file-formats/afl-test/main.c:8
    #8 0x7ff14d6d7b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #9 0x556e3ec85b29 in _start (/root/study/ok-file-formats/afl-test/Asanjpg/Asanjpg+0x2b29)

0x631000011618 is located 9 bytes to the right of 69135-byte region [0x631000000800,0x63100001160f)
allocated by thread T0 here:
    #0 0x7ff14db85b40 in __interceptor_malloc (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xdeb40)
    #1 0x556e3ec85f00 in ok_stdlib_alloc /root/study/ok-file-formats/afl-test/ok_jpg.c:55
    #2 0x556e3ec94b20 in ok_jpg_read_sof /root/study/ok-file-formats/afl-test/ok_jpg.c:1595
    #3 0x556e3ec96ac2 in ok_jpg_decode2 /root/study/ok-file-formats/afl-test/ok_jpg.c:1884
    #4 0x556e3ec97605 in ok_jpg_decode /root/study/ok-file-formats/afl-test/ok_jpg.c:1990
    #5 0x556e3ec868a4 in ok_jpg_read_with_allocator /root/study/ok-file-formats/afl-test/ok_jpg.c:268
    #6 0x556e3ec8671b in ok_jpg_read /root/study/ok-file-formats/afl-test/ok_jpg.c:257
    #7 0x556e3ec85d5e in main /root/study/ok-file-formats/afl-test/main.c:8
    #8 0x7ff14d6d7b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/study/ok-file-formats/afl-test/ok_jpg.c:1054 in ok_jpg_decode_block_progressive
Shadow bytes around the buggy address:
  0x0c627fffa270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c627fffa280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c627fffa290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c627fffa2a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c627fffa2b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c627fffa2c0: 00 07 fa[fa]fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c627fffa2d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c627fffa2e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c627fffa2f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c627fffa300: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c627fffa310: 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
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
```

```
    Stack right redzone:     f3
    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
    Left alloca redzone:     ca
    Right alloca redzone:    cb
==78746==ABORTING
```

Poc

Poc file is here.

Fuzzer & Testcase

Fuzzer is AFL.
Testcase is your testcase in dir ok-file-formats/test/jpg.

brackeen added a commit that referenced this issue on Jun 27, 2020

`Fix issue with miscalculation of block overflow size (#8)` `ce43dd0`

brackeen mentioned this issue on Jun 27, 2020

heap-buffer-overflow in function ok_jpg_decode_block_subsequent_scan() at ok_jpg.c:1102 #7

Closed

brackeen closed this as completed on Jun 27, 2020

---

abergmann commented on Jul 19, 2021

CVE-2020-23707 was assigned to this issue.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants