

[New issue](#)[Jump to bottom](#)

## heap-buffer-overflow in function ok\_jpg\_decode\_block\_subsequent\_scan() at ok\_jpg.c:1102 #7

[Closed](#)

WayneDevMaze opened this issue on Jun 26, 2020 · 2 comments

WayneDevMaze commented on Jun 26, 2020 • edited

## Describe

A heap-buffer-overflow was discovered in ok\_file\_formats. The issue is being triggered in function ok\_jpg\_decode\_block\_subsequent\_scan() at ok\_jpg.c:1102

## 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-1
```

Get ASan reports

```
==98287==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x631000039680 at pc 0x562394639b54 bp 0x7ffee24654e0 sp 0x7ffee24654d0
READ of size 2 at 0x631000039680 thread T0
#0 0x562394639b53 in ok_jpg_decode_block_subsequent_scan /root/study/ok-file-formats/afl-test/ok_jpg.c:1102
#1 0x56239463b11f in ok_jpg_decode_scan /root/study/ok-file-formats/afl-test/ok_jpg.c:1238
#2 0x56239463fc60 in ok_jpg_read_sos /root/study/ok-file-formats/afl-test/ok_jpg.c:1734
#3 0x562394640d3c in ok_jpg_decode2 /root/study/ok-file-formats/afl-test/ok_jpg.c:1900
#4 0x562394641605 in ok_jpg_decode /root/study/ok-file-formats/afl-test/ok_jpg.c:1990
#5 0x5623946398a4 in ok_jpg_read_with_allocator /root/study/ok-file-formats/afl-test/ok_jpg.c:268
#6 0x56239463071b in ok_jpg_read /root/study/ok-file-formats/afl-test/ok_jpg.c:257
#7 0x56239462fd5e in main /root/study/ok-file-formats/afl-test/main.c:8
#8 0x7fe63f9a4b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
#9 0x56239462fb29 in _start (/root/study/ok-file-formats/afl-test/Asanjpg/Asanjpg+0x2b29)
```

0x631000039680 is located 113 bytes to the right of 69135-byte region [0x631000028800,0x63100003960f)
allocated by thread T0 here:

```
#0 0x7fe63fe52b40 in __interceptor_malloc (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xdeb40)
#1 0x56239462ff00 in ok_stdlib_alloc /root/study/ok-file-formats/afl-test/ok_jpg.c:55
#2 0x56239463eb20 in ok_jpg_read_sof /root/study/ok-file-formats/afl-test/ok_jpg.c:1595
#3 0x562394640ac2 in ok_jpg_decode2 /root/study/ok-file-formats/afl-test/ok_jpg.c:1884
#4 0x562394641605 in ok_jpg_decode /root/study/ok-file-formats/afl-test/ok_jpg.c:1990
#5 0x5623946398a4 in ok_jpg_read_with_allocator /root/study/ok-file-formats/afl-test/ok_jpg.c:268
#6 0x56239463071b in ok_jpg_read /root/study/ok-file-formats/afl-test/ok_jpg.c:257
#7 0x56239462fd5e in main /root/study/ok-file-formats/afl-test/main.c:8
#8 0x7fe63f9a4b96 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:1102 in ok\_jpg\_decode\_block\_subsequent\_scan
Shadow bytes around the buggy address:

```
0x0c627ffff280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c627ffff290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c627ffff2a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c627ffff2b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c627ffff2c0: 00 07 fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c627ffff2d0:[fa]fa fa fa fa fa fa fa fa fa fa fa fa
0x0c627ffff2e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c627ffff2f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c627ffff300: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c627ffff310: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c627ffff320: fa 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
==98287==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 invalid component ids (#7)

56fe722

**brackeen** commented on Jun 27, 2020

Owner

Thanks for the issue, and for #8 too.

 **brackeen** closed this as completed on Jun 27, 2020

**abergmann** commented on Jul 19, 2021

[CVE-2020-23706](#) 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

