jasper-1.900.29 Heap Buffer Overflow vulnerability due to a programming mistake

Overview

The vulnerability is similar to the 1st one, including similar root cause.

The vulnerability is found in jasper-1.900.29 (an open-source initiative to provide a free software-based reference implementation of the codec specified in the JPEG-2000 Part-1 standard) using AFL (http://lcamtuf.coredump.cx/afl/). The vulnerability exists in code responsible for decoding the input image to a JP2 file. The vulnerability is a Heap Buffer Overflow vulnerability which can cause Out-of-Bound write due to a programming mistake (i.e. a mistake when setting the size of a memory allocation). The vulnerability can cause Denial-of-Service and may cause Remote-Code-Execution.

Software and Environments

Software: jasper-1.900.29 (http://www.ece.uvic.ca/~frodo/jasper/)

(download link: http://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.29.tar.gz)

Operation System: Ubuntu 16.04.1 i686 LTS

lsb_release –a :

```
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.1 LTS
Release:  16.04
Codename:     xenial
```

uname –a :

```
Linux ubuntu 4.4.0-43-generic #63-Ubuntu SMP Wed Oct 12 13:50:36 UTC 2016 i686 i686 i686 GNU/Linux
```

Compilers: GCC + CLANG + rr

gcc --version :

```
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.2) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

clang --version:

```
clang version 3.8.0-2ubuntu4 (tags/RELEASE_380/final)
Target: i686-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
```

rr --version:

```
rr version 4.4.0
```

Reproduction

GCC Debug

```
cd /*path of jasper-1.900.29 source code*/
mkdir build-debug
cd build-debug
../configure --enable-debug --disable-shared
make
cd src/appl
./jasper -f /*path of PoC2 file*/ -F a.jp2
```

GCC Debug + ASAN

```
cd /*path of jasper-1.900.29 source code*/
mkdir build-debug-asan
cd build-debug-asan
../configure --enable-asan --enable-debug --disable-shared
make
cd src/appl
./jasper –f /*path of PoC2 file*/ -F a.jp2
```

Exception

```
=================================================================
==29945==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x9aafe818 at pc 0x08133ea8 bp 0xbf8d1688 sp 0xbf8d1678
WRITE of size 4 at 0x9aafe818 thread T0
    #0 0x8133ea7 in jpc_qmfb_split_colres ../../../../src/libjasper/jpc/jpc_qmfb.c:548
    #1 0x8137f5b in jpc_ft_analyze ../../../../src/libjasper/jpc/jpc_qmfb.c:1562
    #2 0x810e04b in jpc_tsfb_analyze2 ../../../../src/libjasper/jpc/jpc_tsfb.c:135
    #3 0x810df96 in jpc_tsfb_analyze ../../../../src/libjasper/jpc/jpc_tsfb.c:128
    #4 0x80aeeff in jpc_enc_encodemainbody ../../../../src/libjasper/jpc/jpc_enc.c:1189
    #5 0x80a6711 in jpc_encode ../../../../src/libjasper/jpc/jpc_enc.c:306
    #6 0x8089ad6 in jp2_encode ../../../../src/libjasper/jp2/jp2_enc.c:349
    #7 0x80656c8 in jas_image_encode ../../../../src/libjasper/base/jas_image.c:470
    #8 0x804a70b in main ../../../src/appl/jasper.c:277
    #9 0xb6fcd636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
    #10 0x8049a70  (/home/fire/bing/afl/libraries/jasper/jasper-1.900.29/build-debug-asan/src/appl/jasper+0x8049a70)

0x9aafe818 is located 0 bytes to the right of 4194328-byte region [0x9a6fe800,0x9aafe818)
allocated by thread T0 here:
    #0 0xb7292dee in malloc (/usr/lib/i386-linux-gnu/libasan.so.2+0x96dee)
    #1 0x806fb4c in jas_malloc ../../../../src/libjasper/base/jas_malloc.c:242
    #2 0x806fca8 in jas_alloc2 ../../../../src/libjasper/base/jas_malloc.c:275
    #3 0x8133d95 in jpc_qmfb_split_colres ../../../../src/libjasper/jpc/jpc_qmfb.c:529
    #4 0x8137f5b in jpc_ft_analyze ../../../../src/libjasper/jpc/jpc_qmfb.c:1562
    #5 0x810e04b in jpc_tsfb_analyze2 ../../../../src/libjasper/jpc/jpc_tsfb.c:135
    #6 0x810df96 in jpc_tsfb_analyze ../../../../src/libjasper/jpc/jpc_tsfb.c:128
    #7 0x80aeeff in jpc_enc_encodemainbody ../../../../src/libjasper/jpc/jpc_enc.c:1189
    #8 0x80a6711 in jpc_encode ../../../../src/libjasper/jpc/jpc_enc.c:306
    #9 0x8089ad6 in jp2_encode ../../../../src/libjasper/jp2/jp2_enc.c:349
    #10 0x80656c8 in jas_image_encode ../../../../src/libjasper/base/jas_image.c:470
    #11 0x804a70b in main ../../../src/appl/jasper.c:277
    #12 0xb6fcd636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
```

```
SUMMARY: AddressSanitizer: heap-buffer-overflow ../../../../src/libjasper/jpc/jpc_qmfb.c:548 jpc_qmfb_split_colres
Shadow bytes around the buggy address:
  0x3355fcb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3355fcc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3355fcd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3355fce0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x3355fcf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x3355fd00: 00 00 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa
  0x3355fd10: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x3355fd20: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x3355fd30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x3355fd40: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x3355fd50: 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
==29945==ABORTING
```

From the above, this is a Heap Buffer Overflow vulnerability which can cause Out-of-Bound write.

Analysis

The root cause of this vulnerability is similar to cause of 1st vulnerability.

The out-of-bound write happens in function “**jpc_qmfb_split_colres**”, which can be found in the ASAN exception.

```
=================================================================
==29945==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x9aafe818 at pc 0x08133ea8 bp 0xbf8d1688 sp 0xbf8d1678
WRITE of size 4 at 0x9aafe818 thread T0
    #0 0x8133ea7 in jpc_qmfb_split_colres ../../../../src/libjasper/jpc/jpc_qmfb.c:548
    #1 0x8137f5b in jpc_ft_analyze ../../../../src/libjasper/jpc/jpc_qmfb.c:1562
    #2 0x810e04b in jpc_tsfb_analyze2 ../../../../src/libjasper/jpc/jpc_tsfb.c:135
    #3 0x810df96 in jpc_tsfb_analyze ../../../../src/libjasper/jpc/jpc_tsfb.c:128
    #4 0x80aeeff in jpc_enc_encodemainbody ../../../../src/libjasper/jpc/jpc_enc.c:1189
    #5 0x80a6711 in jpc_encode ../../../../src/libjasper/jpc/jpc_enc.c:306
    #6 0x8089ad6 in jp2_encode ../../../../src/libjasper/jp2/jp2_enc.c:349
    #7 0x80656c8 in jas_image_encode ../../../../src/libjasper/base/jas_image.c:470
    #8 0x804a70b in main ../../../src/appl/jasper.c:277
    #9 0xb6fcd636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
    #10 0x8049a70  (/home/fire/bing/afl/libraries/jasper/jasper-1.900.29/build-debug-asan/src/appl/jasper+0x8049a70)

0x9aafe818 is located 0 bytes to the right of 4194328-byte region [0x9a6fe800,0x9aafe818)
allocated by thread T0 here:
    #0 0xb7292dee in malloc (/usr/lib/i386-linux-gnu/libasan.so.2+0x96dee)
    #1 0x806fb4c in jas_malloc ../../../../src/libjasper/base/jas_malloc.c:242
    #2 0x806fca8 in jas_alloc2 ../../../../src/libjasper/base/jas_malloc.c:275
    #3 0x8133d95 in jpc_qmfb_split_colres ../../../../src/libjasper/jpc/jpc_qmfb.c:529
    #4 0x8137f5b in jpc_ft_analyze ../../../../src/libjasper/jpc/jpc_qmfb.c:1562
    #5 0x810e04b in jpc_tsfb_analyze2 ../../../../src/libjasper/jpc/jpc_tsfb.c:135
    #6 0x810df96 in jpc_tsfb_analyze ../../../../src/libjasper/jpc/jpc_tsfb.c:128
    #7 0x80aeeff in jpc_enc_encodemainbody ../../../../src/libjasper/jpc/jpc_enc.c:1189
    #8 0x80a6711 in jpc_encode ../../../../src/libjasper/jpc/jpc_enc.c:306
    #9 0x8089ad6 in jp2_encode ../../../../src/libjasper/jp2/jp2_enc.c:349
    #10 0x80656c8 in jas_image_encode ../../../../src/libjasper/base/jas_image.c:470
    #11 0x804a70b in main ../../../src/appl/jasper.c:277
    #12 0xb6fcd636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
```

overflow position

allocation position

The related code is

```
511  void jpc_qmfb_split_colres(jpc_fix_t *a, int numrows, int numcols,
512    int stride, int parity)
513  {
514
515      int bufsize = JPC_CEILDIVPOW2(numrows, 1);
516      jpc_fix_t splitbuf[QMFB_SPLITBUFSIZE * JPC_QMFB_COLGRPSIZE];
517      jpc_fix_t *buf = splitbuf;
518      jpc_fix_t *srcptr;
519      jpc_fix_t *dstptr;
520      register jpc_fix_t *srcptr2;
521      register jpc_fix_t *dstptr2;
522      register int n;
523      register int i;
524      int m;
525      int hstartcol;
526
527      /* Get a buffer. */
528      if (bufsize > QMFB_SPLITBUFSIZE) {
529          if (!(buf = jas_alloc2(bufsize, sizeof(jpc_fix_t)))) {
530              /* We have no choice but to commit suicide in this case. */
531              abort();
532          }
533      }
534
535      if (numrows >= 2) {
536          hstartcol = (numrows + 1 - parity) >> 1;
537          // ORIGINAL (WRONG): m = (parity) ? hstartcol : (numrows - hstartcol);
538          m = numrows - hstartcol;
539
540          /* Save the samples destined for the highpass channel. */
541          n = m;
542          dstptr = buf;
543          srcptr = &a[(1 - parity) * stride];
544          while (n-- > 0) {
545              dstptr2 = dstptr;
546              srcptr2 = srcptr;
547              for (i = 0; i < numcols; ++i) {
548                  *dstptr2 = *srcptr2;
549                  ++dstptr2;
550                  ++srcptr2;
551              }
552              dstptr += numcols;
553              srcptr += stride << 1;
554          }
```

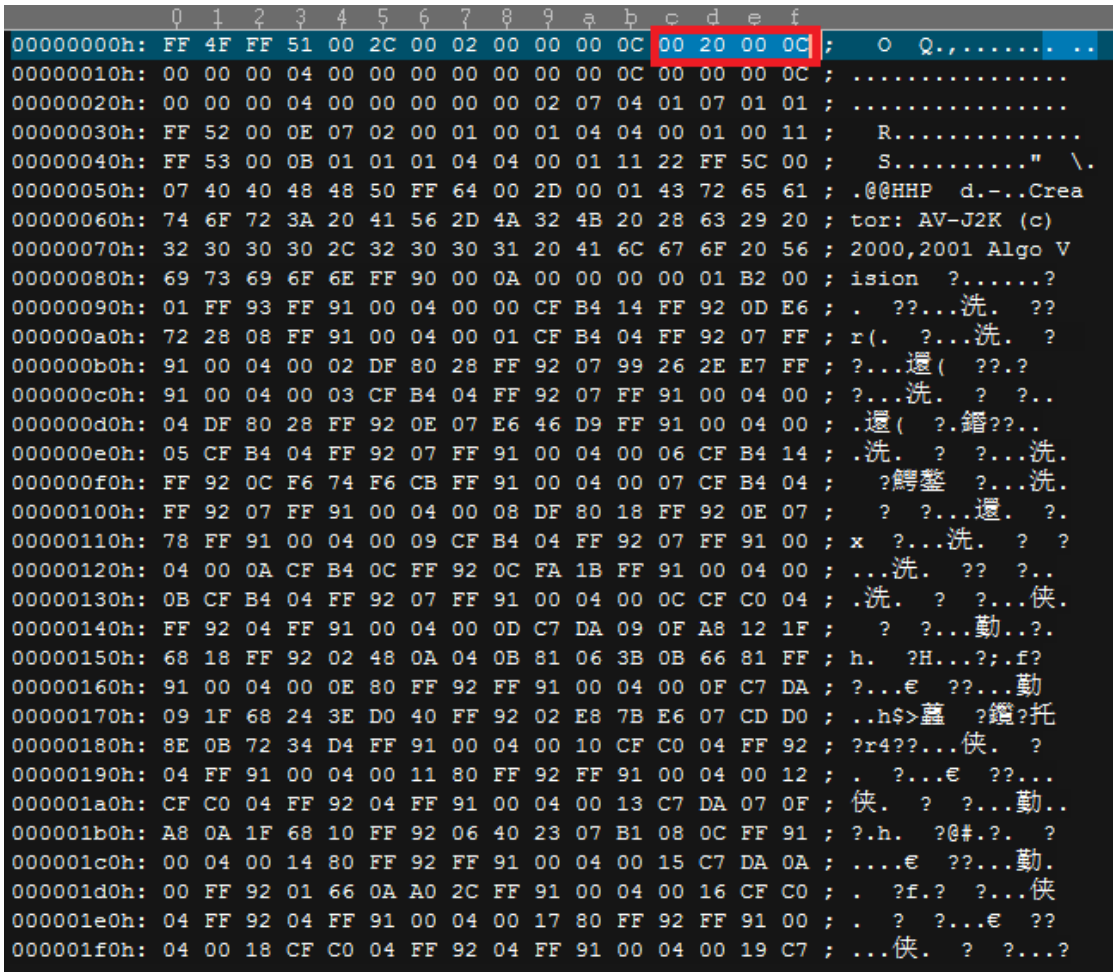
should be

jas_alloc3(bufsize, numcols, sizeof(jpc_fix_t))

Similar to the 1st vulnerability, when “**numrows”** is greater than **QMFB_SPLITBUFSIZE** (i.e. 4096),

the allocated length is "**bufsize = (numrows+1)/2**" while the total write-access length is "**n*numcols = numrows/2**". As a result, out-of-bound write happens in the while loop.

Similarly, value of "**numrows**" can be controlled by modifying a field of the PoC file, i.e. field **@offset 0x0C – offset 0x0F**.



The corresponding debugging info is

```
(gdb) r
Starting program: /home/fire/bing/afl/libraries/jasper/jasper-1.900.29/build-debug-asan/src/appl/jasper -f ../../../../jasper-1.900.28/build-debug/src/appl/outputs-jasper-jp2/crashes/id_000059,sig_11,src_000013,op_flip1,pos_13 -F a.jp2
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".

Breakpoint 1, jpc_qmfb_split_colres (a=0xae9fb800, numrows=2097164, numcols=2, stride=2, parity=0)
    at ../../../../src/libjasper/jpc/jpc_qmfb.c:513
513     {
(gdb) p/x numrows
$1 = 0x20000c
(gdb) c
Continuing.
=================================================================
==30209==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x9b3fe818 at pc 0x08133ea8 bp 0xbffbda38 sp 0xbffbda28
WRITE of size 4 at 0x9b3fe818 thread T0
    #0 0x8133ea7 in jpc_qmfb_split_colres ../../../../src/libjasper/jpc/jpc_qmfb.c:548
    #1 0x8137f5b in jpc_ft_analyze ../../../../src/libjasper/jpc/jpc_qmfb.c:1562
    #2 0x810e04b in jpc_tsfb_analyze2 ../../../../src/libjasper/jpc/jpc_tsfb.c:135
    #3 0x810df96 in jpc_tsfb_analyze ../../../../src/libjasper/jpc/jpc_tsfb.c:128
    #4 0x80aeeff in jpc_enc_encodemainbody ../../../../src/libjasper/jpc/jpc_enc.c:1189
    #5 0x80a6711 in jpc_encode ../../../../src/libjasper/jpc/jpc_enc.c:306
    #6 0x8089ad6 in jp2_encode ../../../../src/libjasper/jp2/jp2_enc.c:349
    #7 0x80656c8 in jas_image_encode ../../../../src/libjasper/base/jas_image.c:470
    #8 0x804a70b in main ../../../src/appl/jasper.c:277
    #9 0xb7832636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
```

In order to trigger the vulnerability, the following condition can be satisfied:

bufsize = (numrows + 1) / 2 > 4096
➔ numrows >= 8193 = 0x2001
➔ file[offset_0x0C, offset_0x0F] >= 0x2001

The vulnerability (out-of-bound write) is very possible to be exploited to achieve Remote-Code-Execution.

Conclusions

This is a Heap Buffer Overflow vulnerability which can cause Out-of-Bound write. I guess the vulnerability is introduced by a programming mistake, i.e. the code should be "**buf = jas_alloc2(bufsize, numcols * sizeof(jpc_fix_t))**" instead of "**buf = jas_alloc2(bufsize, sizeof(jpc_fix_t))**". The vulnerability can cause Denial-of-Service and may cause Remote-Code-Execution.

Patch advice

```
*** jpc_qmfb.c       2016-11-16 23:03:41.000000000 +0800
--- jpc_qmfb_patch2.c    2016-11-25 01:57:14.400086831 +0800
***************
*** 526,532 ****

      /* Get a buffer. */
      if (bufsize > QMFB_SPLITBUFSIZE) {
!         if (!(buf = jas_alloc2(bufsize, sizeof(jpc_fix_t)))) {
              /* We have no choice but to commit suicide in this case. */
              abort();
          }
--- 526,532 ----

      /* Get a buffer. */
      if (bufsize > QMFB_SPLITBUFSIZE) {
!         if (!(buf = jas_alloc3(bufsize, numcols, sizeof(jpc_fix_t)))) {
              /* We have no choice but to commit suicide in this case. */
              abort();
          }
```

Author

name: Bingchang, Liu @VARAS of IIE
org: IIE (http://iie.ac.cn)