

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

Infinite loop in Frame::ParseTrailer #77

✓ Closed chluo911 opened this issue on Jul 26 · 1 comment

chluo911 commented on Jul 26 • edited ▼

version: latest commit [14bf94c](#)poc: [poc.zip](#)

command: ./jpeg poc /dev/null

The backtrace in gdb:

```
(gdb) bt
#0  ByteStream::Get (this=0x790ae0) at bytestream.cpp:223
#1  0x00000000042331e in IOStream::PeekWord (this=0x790ae0)
    at iostream.cpp:543
#2  0x0000000004c38d5 in Frame::ParseTrailer (this=0x792590, io=0x790ae0)
    at frame.cpp:1018
#3  0x00000000043aac9 in JPEG::ReadInternal (this=0x7904c8,
    tags=0x7fffffffdd50) at jpeg.cpp:332
#4  0x00000000043988b in JPEG::Read (this=0x7904c8, tags=0x7fffffffdd50)
    at jpeg.cpp:210
#5  0x00000000041cabb in Reconstruct (infile=<optimized out>,
    outfile=0x7fffffff70c "/dev/null", colortrafo=1, alpha=0x0, upsample=true)
    at reconstruct.cpp:121
#6  0x000000000408b6a in main (argc=<optimized out>, argv=0x790f29)
    at main.cpp:747
```

Root cause:

There is a loop in frame.cpp:1017-1118.

In line 1018, the program continuously reads `marker` in the stream by calling `IOStream::PeekWord()` function.

[libjpeg/marker/frame.cpp](#)

Line 1018 in 842c7ba

```
1018     LONG marker = io->PeekWord();
```

In cases of the `default` branch, the loop won't exit.

[libjpeg/marker/frame.cpp](#)

Lines 1088 to 1117 in 842c7ba

```
1088     if (marker < 0xff00) {
1089         JPG_WARN(MALFORMED_STREAM, "Frame::ParseTrailer",
1090             "expecting a marker or marker segment - stream is out of sync");
1091         // Advance to the next marker and see how it goes from there...
1092         io->Get(); // Remove the invalid thing.
1093         do {
1094             marker = io->Get();
1095         } while(marker != 0xff && marker != ByteStream::EOF);
1096         //
1097         if (marker == ByteStream::EOF) {
```

The problem is that the `IOStream::PeekWord()` function might always return a same value.

Specifically, the `IOStream::PeekWord()` calls `ByteStream::Get()`.

[libjpeg/io/bytestream.cpp](#)

Lines 214 to 224 in 91985dc

```
214     LONG ByteStream::Get(void) // read a single byte (not inlined)
215     {
216
217         if (m_pucBufPtr >= m_pucBufEnd) {
218             if (Fill() == 0) // Found EOF
219                 return EOF;
220         }
221         assert(m_pucBufPtr < m_pucBufEnd);
222
223         return *m_pucBufPtr++;
224     }
```

`IOStream::PeekWord()` returns at line 223 with `m_pucBufPtr++`. However, when using `gdb` to check the value of `m_pucBufPtr`, I found that the `ByteStream::Get()` functions repeatedly read the values from the same address. The `0x790f2a` and `0x790f29` correspond to `byte1` and `byte2` in `IOStream::PeekWord()` and they never change.

```

Breakpoint 1, ByteStream::Get (this=0x790ae0) at bytestream.cpp:223
(gdb) print m_pucBufPtr
$6 = (UBYTE *) 0x790f2a "\310\321\321\321\321\321\312\321\301\321d\001\377\274\350t\321\321", <incomplete sequence \321>
(gdb) c
Continuing.

Breakpoint 1, ByteStream::Get (this=0x790ae0) at bytestream.cpp:223
(gdb) print m_pucBufPtr
$7 = (UBYTE *) 0x790f29 "\377\310\321\321\321\321\321\312\321\301\321d\001\377\274\350t\321\321", <incomplete sequence \321>
(gdb) c
Continuing.

Breakpoint 1, ByteStream::Get (this=0x790ae0) at bytestream.cpp:223
(gdb) print m_pucBufPtr
$8 = (UBYTE *) 0x790f2a "\310\321\321\321\321\321\312\321\301\321d\001\377\274\350t\321\321", <incomplete sequence \321>
(gdb) c
Continuing.

Breakpoint 1, ByteStream::Get (this=0x790ae0) at bytestream.cpp:223
(gdb) print m_pucBufPtr
$9 = (UBYTE *) 0x790f29 "\377\310\321\321\321\321\321\312\321\301\321d\001\377\274\350t\321\321", <incomplete sequence \321>
(gdb) c
Continuing.

```

IOStream::PeekWord() then always returns a same value (calculated by byte1 and byte2). Finally, the program never terminates because the return value forces the program to take the default branch.

thorfdbg commented on Aug 3

Owner

Thank you, this should be fixed in the latest trunk.



thorfdbg closed this as completed on Aug 3

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants

