Null Pointer Dereference Caused Segmentation Fault in gpac/gpac

0

Valid Reported on Mar 25th 2022

Description

Null pointer dereference caused segmentation fault

Proof of Concept

version

```
./bin/gcc/MP4Box -version
MP4Box - GPAC version 2.1-DEV-rev65-g718843df4-master
(c) 2000-2022 Telecom Paris distributed under LGPL v2.1+ - http://gpac.io

Please cite our work in your research:
    GPAC Filters: https://doi.org/10.1145/3339825.3394929
    GPAC: https://doi.org/10.1145/1291233.1291452

GPAC Configuration: --enable-debug
Features: GPAC_CONFIG_LINUX GPAC_64_BITS GPAC_HAS_IPV6 GPAC_HAS_SSL GPAC_HA
```

Command: `./bin/gcc/MP4Box -xmt poc`
Result: `Segmentation fault`
poc
bt:

```
Starting program: /home/ubuntu/fuzz/gpac/bin/gcc/MP4Box -xmt ./poc
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libt
[iso file] Unknown box type url@ in parent dref
```

```
[iso file] Unknown box type mp1Dv in parent stsd
[iso file] extra box maxr found in hinf, deleting
[iso file] Unknown box type t0E18 in parent trak

[iso file] extra box maxr found in hinf, deleting
[iso file] Unknown box type 80rak in parent moov
[iso file] Incomplete box mdat - start 11495 size 803701
[iso file] Incomplete file while reading for dump - aborting parsing
[iso file] Unknown box type url@ in parent dref
[iso file] Unknown box type mp1Dv in parent stsd
[iso file] extra box maxr found in hinf, deleting
[iso file] Unknown box type t0E18 in parent trak
[iso file] extra box maxr found in hinf, deleting
[iso file] Unknown box type 80rak in parent moov
[iso file] Incomplete box mdat - start 11495 size 803701
[iso file] Incomplete file while reading for dump - aborting parsing
MPEG-4 BIFS Scene Parsing
[ODF] Reading bifs config: shift in sizes (not supported)

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7240bc7 in BS_ReadByte (bs=0x5555557d29b0) at utils/bitstream.c:
362         res = bs->original[bs->position++];
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
```

```
 RAX  0x0
 RBX  0x0
 RCX  0x1
 RDX  0x5555557d29b0 ◂— 0x0
 RDI  0x5555557d29b0 ◂— 0x0
 RSI  0x0
 R8   0x0
 R9   0x0
 R10  0x9
 R11  0x7ffff72aa158 (gf_node_get_graph) ◂— push   rbp
 R12  0x5555557d4890 —▸ 0x5555557d4460 ◂— 0x0
 R13  0x5555557d4460 ◂— 0x0
 R14  0x1
 R15  0x0
 RBP  0x7fffffff8730 —▸ 0x7fffffff8760 —▸ 0x7fffffff8790 —▸ 0x7fffffff87c0
 RSP  0x7fffffff8700 ◂— 0x0
 RIP  0x7ffff7240bc7 (BS_ReadByte+166) ◂— movzx  eax, byte ptr [rax]
```

```
 ► 0x7ffff7240bc7 <BS_ReadByte+166>    movzx  eax, byte ptr [rax]
   0x7ffff7240bca <BS_ReadByte+169>    mov    byte ptr [rbp - 0x13], al

   0x7ffff7240bcd <BS_ReadByte+172>    mov    rax, qword ptr [rbp - 0x28]
   0x7ffff7240bd1 <BS_ReadByte+176>    mov    eax, dword ptr [rax + 0x50]
   0x7ffff7240bd4 <BS_ReadByte+179>    test   eax, eax
   0x7ffff7240bd6 <BS_ReadByte+181>    je     BS_ReadByte+336
    ↓
   0x7ffff7240c71 <BS_ReadByte+336>    movzx  eax, byte ptr [rbp - 0x13]
   0x7ffff7240c75 <BS_ReadByte+340>    jmp    BS_ReadByte+896
    ↓
   0x7ffff7240ea1 <BS_ReadByte+896>    mov    rcx, qword ptr [rbp - 8]
   0x7ffff7240ea5 <BS_ReadByte+900>    xor    rcx, qword ptr fs:[0x28]
   0x7ffff7240eae <BS_ReadByte+909>    je     BS_ReadByte+916
```

```
In file: /home/ubuntu/fuzz/gpac/src/utils/bitstream.c
   357      if (bs->position >= bs->size) {
   358          if (bs->EndOfStream) bs->EndOfStream(bs->par);
   359          if (!bs->overflow_state) bs->overflow_state = 1;
   360          return 0;
   361      }
 ► 362      res = bs->original[bs->position++];
   363
   364      if (bs->remove_emul_prevention_byte) {
   365          if ((bs->nb_zeros==2) && (res==0x03) && (bs->position<bs->s
   366              bs->nb_zeros = 0;
   367              res = bs->original[bs->position++];
```

```
00:0000| rsp 0x7fffffff8700 ◂— 0x0
01:0008|     0x7fffffff8708 —▸ 0x5555557d29b0 ◂— 0x0
02:0010|     0x7fffffff8710 ◂— 0x0
... ↓        2 skipped
05:0028|     0x7fffffff8728 ◂— 0xe537f499f4c1e900
06:0030| rbp 0x7fffffff8730 —▸ 0x7fffffff8760 —▸ 0x7fffffff8790 —▸ 0x7ffff
07:0038|     0x7fffffff8738 —▸ 0x7ffff7240edb (gf_bs_read_bit+36) ◂— movzx
```

```
 ► f 0   0x7ffff7240bc7 BS_ReadByte+166
   f 1   0x7ffff7240edb gf_bs_read_bit+36
   f 2   0x7ffff7240f67 gf_bs_read_int+64
   f 3   0x7ffff73d9f47 BM_ParseCommand+87
```

```
f 4   0x7ffff73da181 gf_bifs_flush_command_list+296
f 5   0x7ffff73da561 gf_bifs_decode_command_list+339
f 6   0x7ffff75c1189 gf_sm_load_run_isom+1987

f 7   0x7ffff75a3123 gf_sm_load_run+42
```

```
pwndbg> where
#0  0x00007ffff7240bc7 in BS_ReadByte (bs=0x5555557d29b0) at utils/bitstrea
#1  0x00007ffff7240edb in gf_bs_read_bit (bs=0x5555557d29b0) at utils/bitst
#2  0x00007ffff7240f67 in gf_bs_read_int (bs=0x5555557d29b0, nBits=1) at ut
#3  0x00007ffff73d9f47 in BM_ParseCommand (codec=0x5555557def00, bs=0x5555
#4  0x00007ffff73da181 in gf_bifs_flush_command_list (codec=0x5555557def00)
#5  0x00007ffff73da561 in gf_bifs_decode_command_list (codec=0x5555557def00
#6  0x00007ffff75c1189 in gf_sm_load_run_isom (load=0x7fffffff89c0) at scen
#7  0x00007ffff75a3123 in gf_sm_load_run (load=0x7fffffff89c0) at scene_man
#8  0x0000555555582280 in dump_isom_scene ()
#9  0x00005555555789de in mp4boxMain ()
#10 0x00007ffff6dcdbf7 in __libc_start_main (main=0x55555556c140 <main>, ar
#11 0x000055555556c17a in _start ()
```

After Debugging,I found
src/bifs/memory_decoder.c:953

```
GF_Err gf_bifs_flush_command_list(GF_BifsDecoder *codec)
```

```
            if (cbi->cb->bufferSize) {
                bs = gf_bs_new((char*)cbi->cb->buffer, cbi->cb->bufferSize,
                gf_bs_set_eos_callback(bs, BM_EndOfStream, codec);
                e = BM_ParseCommand(codec, bs, cbi->cb->commandList);
                gf_bs_del(bs);
            }
```

after called `gf_bs_new` , sets bs->origin to null
then using `BM_ParseCommand` function
src/bifs/memory_decoder.c:887

```
GF_Err BM_ParseCommand(GF_BifsDecoder *codec, GF_BitStream *bs, GF_List *cc
{
```

```
{
    u8 go, type;
    GF_Err e;

    go = 1;
    e = GF_OK;
    GF_SceneGraph *cur_graph = codec->current_graph;
    GF_Proto *cur_proto = codec->pCurrentProto;

    codec->LastError = GF_OK;
    while (go) {
        type = gf_bs_read_int(bs, 2);
        switch (type) {
        case 0:
            e = BM_ParseInsert(codec, bs, com_list);
            break;
        case 1:
            e = BM_ParseDelete(codec, bs, com_list);
            break;
        case 2:
            e = BM_ParseReplace(codec, bs, com_list);
            break;
        case 3:
            e = BM_SceneReplace(codec, bs, com_list);
            break;
        }
        if (e) break;
        go = gf_bs_read_int(bs, 1);
    }
    while (gf_list_count(codec->QPs)) {
        gf_bifs_dec_qp_remove(codec, GF_TRUE);
    }

    codec->current_graph = cur_graph;
    codec->pCurrentProto = cur_proto;
    return e;
}
```

in `type = gf_bs_read_int(bs, 2);`

src/utils/bitstream.c

src/utils/bitstream.c

```
GF_EXPORT
u32 gf_bs_read_int(GF_BitStream *bs, u32 nBits)
{
    u32 ret;
    bs->total_bits_read+= nBits;

#ifndef NO_OPTS
    if (nBits + bs->nbBits <= 8) {
        bs->nbBits += nBits;
        ret = (bs->current >> (8 - bs->nbBits) ) & bits_mask[nBits];
        return ret;
    }
#endif
    ret = 0;
    while (nBits-- > 0) {
        ret <<= 1;
        ret |= gf_bs_read_bit(bs);
    }
    return ret;
}
```

In `gf_bs_read_bit` using `BS_ReadByte` function

```
GF_EXPORT
u8 gf_bs_read_bit(GF_BitStream *bs)
{
    if (bs->nbBits == 8) {
        bs->current = BS_ReadByte(bs);
        bs->nbBits = 0;
    }
#ifdef NO_OPTS
    {
        s32 ret;
        bs->current <<= 1;
        bs->nbBits++;
        ret = (bs->current & 0x100) >> 8;
```

```
        return (u8) ret;
    }
#else

    return (u8) (bs->current & bit_mask[bs->nbBits++]) ? 1 : 0;
#endif

}
```

In src/utils/bitstream.c `BS_ReadByte`

```
/*fetch a new byte in the bitstream switch between packets*/
static u8 BS_ReadByte(GF_BitStream *bs)
{
    Bool is_eos;
    if (bs->bsmode == GF_BITSTREAM_READ) {
        u8 res;
        if (bs->position >= bs->size) {
            if (bs->EndOfStream) bs->EndOfStream(bs->par);
            if (!bs->overflow_state) bs->overflow_state = 1;
            return 0;
        }
        res = bs->original[bs->position++];

        if (bs->remove_emul_prevention_byte) {
            if ((bs->nb_zeros==2) && (res==0x03) && (bs->position<bs->size)
                bs->nb_zeros = 0;
                res = bs->original[bs->position++];
            }
            if (!res) bs->nb_zeros++;
            else bs->nb_zeros = 0;
        }
        return res;
    }
    if (bs->cache_write)
        bs_flush_write_cache(bs);

    is_eos = gf_feof(bs->stream);

    /*we are in FILE mode, test for end of file*/
    if (!is_eos || bs->cache_read) {
```

```
        u8 res;
        Bool loc_eos=GF_FALSE;
        assert(bs->position<=bs->size);
        bs->position++;

        res = gf_bs_load_byte(bs, &loc_eos);
        if (loc_eos) goto bs_eof;

        if (bs->remove_emul_prevention_byte) {
            if ((bs->nb_zeros==2) && (res==0x03) && (bs->position<bs->size)
                u8 next = gf_bs_load_byte(bs, &loc_eos);
                if (next < 0x04) {
                    bs->nb_zeros = 0;
                    res = next;
                    bs->position++;
                } else {
                    gf_bs_seek(bs, bs->position);
                }
            }
            if (!res) bs->nb_zeros++;
            else bs->nb_zeros = 0;
        }
        return res;
    }

bs_eof:
    if (bs->EndOfStream) {
        bs->EndOfStream(bs->par);
        if (!bs->overflow_state) bs->overflow_state = 1;
    } else {
        GF_LOG(GF_LOG_ERROR, GF_LOG_CORE, ("[BS] Attempt to overread bitstr
    }
    assert(bs->position <= 1+bs->size);
    return 0;
}
```

`res = bs->original[bs->position++];` Here using bs->original which is a nu

dereference this null pointer caused segmentation fault

it's very similar to https://huntr.dev/bounties/851942a4-1d64-4553-8fdc-9fccd167864b/

It's very similar to https://huntr.dev/bounties/...

CVE

CVE-2022-1172
(Published)

Vulnerability Type
CWE-476: NULL Pointer Dereference

Severity
Medium (5.6)

Visibility
Public

Status
Fixed

Found by

Joe1sn
@joe1sn
unranked

This report was seen 631 times.

We are processing your report and will contact the **gpac** team within 24 hours. 8 months ago

We have contacted a member of the **gpac** team and are waiting to hear back 8 months ago

A **gpac/gpac** maintainer 8 months ago

https://github.com/gpac/gpac/issues/2153

A **gpac/gpac** maintainer validated this vulnerability 8 months ago

Joe1sn has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

A **gpac/gpac** maintainer marked this as fixed in **2.1.0-DEV** with commit **55a...**

The fix bounty has been dropped ✖

This vulnerability will not receive a CVE ✖

Joe1sn 8 months ago Researcher

@maintainer @admin it seems like I forgot to write the Impact,this vulnerablility is capable of crashing software, so i think it's can described as Dos.may i have CVE assigned in this case?

Jamie Slome 8 months ago Admin

Sure, we can assign a CVE here. We do require the permission of the maintainer before we proceed with this.

@maintainer - are you happy for us to assign and publish a CVE for this report?

A gpac/gpac maintainer 8 months ago

Yes, please do whatever is the best practice.

Jamie Slome 8 months ago Admin

Sorted! 👍

Sign in to join this conversation