Heap buffer overflow in isom_hinter.c:766 in gf_hinter_track_process() #1479

Closed **14isnot40** opened this issue on May 12, 2020 · 2 comments

14isnot40 commented on May 12, 2020 • edited

- [y] I looked for a similar issue and couldn't find any.
- [y] I tried with the latest version of GPAC. Installers available at http://gpac.io/downloads/gpac-nightly-builds/
- [y] I give enough information for contributors to reproduce my issue (meaningful title, github labels, platform and compiler, command-line ...). I can share files anonymously with this dropbox: https://www.mediafire.com/filedrop/filedrop_hosted.php?drop=eec9e058a9486fe4e99c33021481d9e1826ca9dbc242a6cfaab0fe95da5e5d95

Describe the bug
A heap-based buffer overflow was discovered in libgpac, during the pointer ptr points to the wrong memory area operation. The issue is being triggered in the function gf_hinter_track_process() at isom_hinter_track_process.c.

To Reproduce
Steps to reproduce the behavior:

1. Compile gpac according to the default configuration

```
./configure --extra-cflags="-fsanitize=address,undefined -g" --extra-ldflags="-fsanitize=address,undefined -ldl -g"
```

2. execute command

```
MP4Box -hint $poc
```

poc can be found here.

Expected behavior
An attacker can exploit this vulnerability by submitting a malicious media file that exploits this issue. This will result in a Denial of Service (DoS) and potentially Information Exposure when the application attempts to process the file.

Screenshots
ASAN Reports

```
==32436==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000e7f9 at pc 0x7ffff44178c2 bp 0x7fffffff8de0 sp 0x7fffffff8dd0
READ of size 1 at 0x60200000e7f9 thread T0
    #0 0x7ffff44178c1 in gf_hinter_track_process (/usr/local/lib/libgpac.so.8+0x24ce8c1)
    #1 0x40e68c in HintFile (/usr/local/bin/MP4Box+0x40e68c)
    #2 0x419db6 in mp4boxMain (/usr/local/bin/MP4Box+0x419db6)
    #3 0x7ffff1b9f82f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #4 0x40dc18 in _start (/usr/local/bin/MP4Box+0x40dc18)

0x60200000e7f9 is located 0 bytes to the right of 9-byte region [0x60200000e7f0,0x60200000e7f9)
allocated by thread T0 here:
    #0 0x7ffff6f02602 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x98602)
    #1 0x7ffff3f83fb8 in Media_GetSample (/usr/local/lib/libgpac.so.8+0x203afb8)

SUMMARY: AddressSanitizer: heap-buffer-overflow ??:0 gf_hinter_track_process
Shadow bytes around the buggy address:
  0x0c047fff9ca0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9cb0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9cc0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9cd0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9ce0: fa fa fa fa fa fa fa fa fa fa 00 00 fa fa 00 00
=>0x0c047fff9cf0: fa fa 00 00 fa fa 00 00 fa fa 00 00 fa fa 00[01]
  0x0c047fff9d00: fa fa fd fd fa fa fd fd fa fa fd fd fa fa fd fd
  0x0c047fff9d10: fa fa fd fd fa fa fd fd fa fa fd fd fa fa fd fd
  0x0c047fff9d20: fa fa fd fd fa fa fd fd fa fa fd fd fa fa fd fd
  0x0c047fff9d30: fa fa fd fd fa fa fd fd fa fa fd fd fa fa fd fd
  0x0c047fff9d40: fa fa fd fd fa fa fd fd fa fa 04 fa fa fa fd fd
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
==32436==ABORTING
```

Possible causes of vulnerabilities is in the function gf_hinter_track_process() at isom_hinter_track_process.c.

```
              while (remain) {
                  size = 0;
                  v = tkHint->avc_nalu_size;
                  while (v) {
                      size |= (u8) *ptr;
                      ptr++;
```

```
            remain--;
            v-=1;
            if (v) size<<=8;
        }
```

System (please complete the following information):

- OS version : Ubuntu 16.04
- GPAC Version : GPAC 0.8.0-e10d39d-master branch

jeanlf commented on Jun 11, 2020 Contributor

this has been fixed by fixing your related bugs, thanks for the report

jeanlf closed this as completed on Jun 11, 2020

carnil commented on Sep 27, 2021

Bisecting the issue leads to `b286aa0`

```
# broken: [e4ed32bf56fc02fb8a04b9e13f4d7bdae2b3ae12] fixed potential crash in traf merging when packed samples are used
git bisect broken e4ed32bf56fc02fb8a04b9e13f4d7bdae2b3ae12
# fixed: [47d8bc5b3ddeed6d775197ebefae7c94a45d9bf2] fixed potential crashes on broken fragmented files - cf #1481 and #1480
git bisect fixed 47d8bc5b3ddeed6d775197ebefae7c94a45d9bf2
# broken: [bcfd53a601a66a9e39f89c697af5bc3b355389b2] fixed potential bug reading fragmented file stats
git bisect broken bcfd53a601a66a9e39f89c697af5bc3b355389b2
# broken: [39367c29f21232e61f6883607c1d1c677bc28ccd] fixed bugs introduced by 211ab52d
git bisect broken 39367c29f21232e61f6883607c1d1c677bc28ccd
# broken: [822fba627b3e5fb29cb29af94a0c6735c82d1a90] fixed potential crash - cf #1487
git bisect broken 822fba627b3e5fb29cb29af94a0c6735c82d1a90
# broken: [4af6987d4d08bb88ca4149d94a2708a4ed6fa8c0] fixed potential crash - cf #1485
git bisect broken 4af6987d4d08bb88ca4149d94a2708a4ed6fa8c0
# fixed: [b286aa0cdc0cb781e96430c8777d38f066a2c9f9] fixed potential crash - cf #1483
git bisect fixed b286aa0cdc0cb781e96430c8777d38f066a2c9f9
# first fixed commit: [b286aa0cdc0cb781e96430c8777d38f066a2c9f9] fixed potential crash - cf #1483
```

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants