Heap buffer overflow in Ap4TrunAtom.cpp when running mp42aac #415

Open 5hadowblad3 opened this issue on Aug 9, 2019 · 0 comments

Assignees

Labels fuzzing

5hadowblad3 commented on Aug 9, 2019

There is a heap buffer overflow in Ap4TrunAtom.cpp when running mp42aac.
Distributor ID: Ubuntu
Description: Ubuntu 16.04.6 LTS
Release: 16.04
Codename: xenial
gcc: 5.4.0

To reproduce the bug,
compile the project with flag
'-DCMAKE_C_FLAGS=-g -m32 -fsanitize=address,undefined'

then run:
'./mp42aac input /dev/null'

```
==147243==ERROR: AddressSanitizer: heap-buffer-overflow on address 0xf4208b40 at pc 0x083eb6d5 bp 0xffef35d8 sp 0xffef35c8
WRITE of size 4 at 0xf4208b40 thread T0
#0 0x83eb6d4 in AP4_Array<AP4_TrunAtom::Entry>::SetItemCount(unsigned int) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4TrunAtom.h:58
#1 0x83d7d9b in AP4_TrunAtom::AP4_TrunAtom(unsigned int, unsigned char, unsigned int, AP4_ByteStream&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4TrunAtom.cpp:127
#2 0x83dde35 in AP4_TrunAtom::Create(unsigned int, AP4_ByteStream&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4TrunAtom.cpp:51
#3 0x82dd3b4 in AP4_AtomFactory::CreateAtomFromStream(AP4_ByteStream&, unsigned int, unsigned int, unsigned long long, AP4_Atom*&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4AtomFactory.cpp:408
#4 0x8301ca3 in AP4_AtomFactory::CreateAtomFromStream(AP4_ByteStream&, unsigned long long&, AP4_Atom*&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4AtomFactory.cpp:225
#5 0x82b6bae in AP4_ContainerAtom::ReadChildren(AP4_AtomFactory&, AP4_ByteStream&, unsigned long long) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4ContainerAtom.cpp:194
#6 0x82b6bae in AP4_ContainerAtom::AP4_ContainerAtom(unsigned int, unsigned long long, bool, AP4_ByteStream&, AP4_AtomFactory&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4ContainerAtom.cpp:139
#7 0x841a898 in AP4_MoovAtom::AP4_MoovAtom(unsigned int, AP4_ByteStream&, AP4_AtomFactory&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4MoovAtom.cpp:80
#8 0x82e2631 in AP4_MoovAtom::Create(unsigned int, AP4_ByteStream&, AP4_AtomFactory&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4MoovAtom.h:56
#9 0x82e2631 in AP4_AtomFactory::CreateAtomFromStream(AP4_ByteStream&, unsigned int, unsigned int, unsigned long long, AP4_Atom*&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4AtomFactory.cpp:363
#10 0x82fa1f7 in AP4_AtomFactory::CreateAtomFromStream(AP4_ByteStream&, unsigned long long&, AP4_Atom*&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4AtomFactory.cpp:225
#11 0x82fa1f7 in AP4_AtomFactory::CreateAtomFromStream(AP4_ByteStream&, AP4_Atom*&) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4AtomFactory.cpp:151
#12 0x809a044 in AP4_File::ParseStream(AP4_ByteStream&, AP4_AtomFactory&, bool) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4File.cpp:104
#13 0x809a044 in AP4_File::AP4_File(AP4_ByteStream&, bool) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4File.cpp:78
#14 0x8082ce7 in main /mnt/data/playground/mp42-a/Source/C++/Apps/Mp42Aac/Mp42Aac.cpp:250
#15 0xf6a26636 in __libc_start_main (/lib/i386-linux-gnu/libc.so.6+0x18636)
#16 0x808df1b (/mnt/data/playground/mp42-patch/Build/mp42aac+0x808df1b)

0xf4208b40 is located 0 bytes to the right of 34624-byte region [0xf4200400,0xf4208b40)
allocated by thread T0 here:
#0 0xf729dcd6 in operator new(unsigned int) (/usr/lib32/libasan.so.2+0x97cd6)
#1 0x83e9fa7 in AP4_Array<AP4_TrunAtom::Entry>::EnsureCapacity(unsigned int) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4Array.h:172
#2 0x83e9fa7 in AP4_Array<AP4_TrunAtom::Entry>::SetItemCount(unsigned int) /mnt/data/playground/mp42-a/Source/C++/Core/Ap4Array.h:210
```

```
SUMMARY: AddressSanitizer: heap-buffer-overflow /mnt/data/playground/mp42-a/Source/C++/Core/Ap4TrunAtom.h:58 AP4_Array<AP4_TrunAtom::Entry>::SetItemCount(unsigned int)
Shadow bytes around the buggy address:
0x3e841110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x3e841120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x3e841130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x3e841140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x3e841150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x3e841160: 00 00 00 00 00 00 00 00[fa]fa fa fa fa fa fa fa
0x3e841170: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x3e841180: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x3e841190: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x3e8411a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x3e8411b0: 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
==147243==ABORTING
```

poc_input5.zip

barbibulle self-assigned this on Aug 25, 2019

barbibulle added the **fuzzing** label on Aug 25, 2019

Assignees

barbibulle

Labels

fuzzing

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants