New issue

Jump to bottom

[Security]memory leak with MP4Box #1705

Closed **5n1p3r0010** opened this issue on Mar 11, 2021 · 1 comment

5n1p3r0010 commented on Mar 11, 2021 • edited

Hi,

There is an info leak issue with MP4Box,this can reproduce on the lattest commit aka ee87237 .The security impact of this issue is the attacker can use this to bypass some system security feature like aslr or stack canary(guard).

Steps to reproduce

build with asan:

```
CC=gcc CXX=g++ CFLAGS="-fsanitize=address" CXXFLAGS="-fsanitize=address" LDFLAGS="-fsanitize=address" ./configure
make
```

run as:

```
MP4Box -hint <poc> -out /dev/null
```

shows the following log:

```
r00t@5n1p3r0010:~/fuzz/target/gpac/bin/gcc$ ./MP4Box -hint /mnt/hgfs/pwn_share/crash/OpenSourceProjects/gpac/triage/memleak/12 -out /dev/null
[iso file] Unknown box type dr88f in parent dinf
[iso file] Missing dref box in dinf
[iso file] Box "stsz" (start 1151) has 36 extra bytes
[iso file] Box "stco" (start 1219) has 4 extra bytes
Hinting track ID 1 - Type "avc1:avc1" (H264) - BW 4296563 kbps
Error while hinting (IsoMedia File is truncated)

        Error: IsoMedia File is truncated

=================================================================
==6348==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 48 byte(s) in 1 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac68efb4f in gf_odf_avc_cfg_new (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0x271b4f)

Indirect leak of 160 byte(s) in 2 object(s) allocated from:
    #0 0x7ffac6f9fffe in __interceptor_realloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dffe)
    #1 0x7ffac67221d5 in gf_list_add (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0xa41d5)

Indirect leak of 36 byte(s) in 1 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac6854474 in AVC_DuplicateConfig (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0x1d6474)

Indirect leak of 32 byte(s) in 2 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac6722131 in gf_list_new (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0xa4131)

Indirect leak of 24 byte(s) in 1 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac685445c in AVC_DuplicateConfig (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0x1d645c)

Indirect leak of 24 byte(s) in 1 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac68544cc in AVC_DuplicateConfig (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0x1d64cc)

Indirect leak of 5 byte(s) in 1 object(s) allocated from:
    #0 0x7ffac6f9fbc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x7ffac68544e4 in AVC_DuplicateConfig (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0x1d64e4)

SUMMARY: AddressSanitizer: 329 byte(s) leaked in 9 allocation(s).
```

Analyze

To debug this,we'd better build using the following config:

```
CFLAGS="-g -fsanitize=address" CXXFLAGS="-g -fsanitize=address" ./configure --enable-debug
make
```

using gdb to debug this

```
file MP4Box
set args -hint <poc> -out /dev/null
b gf_odf_avc_cfg_new
```

To my analyze,these info leaks are due to the malloc/realloc size are not properly checked,take the

```
Indirect leak of 160 byte(s) in 2 object(s) allocated from:
    #0 0x7ffac6f9fffe in __interceptor_realloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dffe)
    #1 0x7ffac67221d5 in gf_list_add (/home/r00t/fuzz/target/gpac/bin/gcc/libgpac.so.10+0xa41d5)
```

for example.

gf_list_add was called in avc_ext.c:AVC_DuplicateConfig,this routine try to realloc memory based on the given pointer,but the size of the realloc memory wasn't checked in 'realloc_chain' function,the stack trace was:

```
gf_odf_avc_cfg_new => gf_list_add => realloc_chain
```

when we set the breakpoint on gf_odf_avc_cfg_new in gdb,when the third time calls it,we can see the realloc size was 32767:

```
pwndbg> print ptr
$15 = (GF_List *) 0x7fffffff90c0
pwndbg> print *(GF_List *) 0x7fffffff90c0
$16 = {
  slots = 0x7fffffff9100,
  entryCount = 4152679278,
  allocSize = 32767
}
```

if the realloc size was big enough like this,though the MP4Box may fail and exit here,but the attacker can still leak some memory based on the given realloc pointer,this can bypass some system security feature like aslr or stack canary,combined with other bugs this can trigger arbitrary code execution.

To fix these info leaks,we should check these malloc/realloc size more carefully.

Reporter

5n1p3r0010 from Topsec Alpha Lab

memleak1.zip

5n1p3r0010 commented on Mar 12, 2021 Author

Also,for the effenciency the system won't realease a chunk when you free it,but put it in some free lists,and reuse it according to some strategy when you re malloc/realloc it.So in some complex routine when we malloc a chunk we may get the previous freed chunk,and the data in the newly malloced chunk probably remains as previous(which may contain some libc address or previous structure data),so right way is to initialize the data(maybe to 0) in the newly malloced chunk before use it.

Absolutely,this initializing newly created chunk data routine can't be seen in the reported code snippet(just initialize some structure data),which will also leak memory.

jeanlf closed this as completed in 2da2f68 on Mar 12, 2021

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant