Talos Vulnerability Report

TALOS-2021-1376

AnyCubic Chitubox AnyCubic Plugin readDatHeadVec heap-based buffer overflow vulnerability

JANUARY 10, 2022

CVE NUMBER

CVE-2021-21948

Summary

A heap-based buffer overflow vulnerability exists in the readDatHeadVec functionality of AnyCubic Chitubox AnyCubic Plugin 1.0.0. A specially-crafted GF file can lead to a heap buffer overflow. An attacker can provide a malicious file to trigger this vulnerability.

Tested Versions

AnyCubic Chitubox AnyCubic Plugin 1.0.0
Chitubox Basic V1.8.1

Product URLs

https://www.chitubox.com

CVSSv3 Score

7.8 - CVSS:3.0/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

CWE

CWE-122 - Heap-based Buffer Overflow

Details

The Chitubox AnyCubic plugin is used to convert the output of the Chitubox slicer (general format files) into the format expected by AnyCubic's series of printers. These converted files are then used directly for all functionality provided by the printers.

A heap buffer overflow occurs within the `GfFile::readDatHeadVec` function seen below. The overflow occurs due to an integer overflow that occurs at `[1]` . This overflow is caused by using 32-bit registers instead of the extended 64-bit registers. The `imul` instruction will truncate the value during the multiplication, losing the most significant 32 bits. This calculated sized is used at `[2]` to allocate the correct size for the vector of `GfDatHead_t`. At `[3]` a very similar multiplication occurs, but uses a 64-bit register for the multiplication instead, thus eliminating the possibility of an overflow (since both values are loaded as 32-bit values). This new value calculated at `[3]` is used at `[4]` in the `fread` as a length of data to read into the buffer sized using the value calculated at `[1]` which is too small, resulting in a buffer overflow while reading the file contents into the buffer.

```
00006630  int64_t GfFile::readDatHeadVec(struct GfFile* this)

00006630  f30f1efa          endbr64
00006634  53                push    rbx {__saved_rbx}
00006635  488b9728010000    mov     rdx, qword [rdi+0x128 {GfFile::gfDataHead.end}]
0000663c  4889fb            mov     rbx, rdi
0000663f  488b8f20010000    mov     rcx, qword [rdi+0x120 {GfFile::gfDataHead.begin}]
00006646  // Load 0x6
00006646  8b7734            mov     esi, dword [rdi+0x34 {GfFile::headerBuffer.__offset(0x24).d}]
00006649  4889d0            mov     rax, rdx
0000664c  // Multiply by 0x80000001
0000664c  //
0000664c  // This is an overflow
0000664c  0faf7730          imul    esi, dword [rdi+0x30 {GfFile::headerBuffer.__offset(0x20).d}]             [1]
00006650  48bf398ee3388ee3…mov     rdi, 0x8e38e38e38e38e39
0000665a  4829c8            sub     rax, rcx
0000665d  48c1f802          sar     rax, 0x2
00006661  480fafc7          imul    rax, rdi
00006665  4863f6            movsxd  rsi, esi
00006668  4839c6            cmp     rsi, rax
0000666b  7753              ja      0x66c0

0000666d  7314              jae     0x6683

0000666f  488d04f6          lea     rax, [rsi+rsi*8]
00006673  488d0481          lea     rax, [rcx+rax*4]
00006677  4839c2            cmp     rdx, rax
0000667a  7407              je      0x6683

0000667c  48898328010000    mov     qword [rbx+0x128 {GfFile::gfDataHead.end}], rax

00006683  48637338          movsxd  rsi, dword [rbx+0x38 {GfFile::headerBuffer.datHeadVecOffset}]
00006687  488b7b08          mov     rdi, qword [rbx+0x8 {GfFile::GfFilePointer}]
0000668b  31d2              xor     edx, edx  {0x0}
0000668d  e87ebcffff        call    fseek
00006692  // Load 0xffffffff80000001
00006692  48635330          movsxd  rdx, dword [rbx+0x30 {GfFile::headerBuffer.__offset(0x20).d}]
00006696  // Load 0x6
00006696  48634334          movsxd  rax, dword [rbx+0x34 {GfFile::headerBuffer.__offset(0x24).d}]
0000669a  be01000000        mov     esi, 0x1
0000669f  488b4b08          mov     rcx, qword [rbx+0x8 {GfFile::GfFilePointer}]
000066a3  488bbb20010000    mov     rdi, qword [rbx+0x120 {GfFile::gfDataHead.begin}]
000066aa  // Same multiply as earlier, but with 64-bit registers
000066aa  480fafc2          imul    rax, rdx                                                                  [3]
000066ae  488d14c0          lea     rdx, [rax+rax*8]
000066b2  48c1e202          shl     rdx, 0x2
000066b6  // This fread will read in the un-overflowed value of bytes into a buffer only sized for
000066b6  // the overflowed 32 bit value
000066b6  e8d5bcffff        call    fread                                                                     [4]
000066bb  31c0              xor     eax, eax  {0x0}
000066bd  5b                pop     rbx {__saved_rbx}
000066be  c3                retn     {__return_addr}
000066c0  4829c6            sub     rsi, rax
000066c3  488dbb20010000    lea     rdi, [rbx+0x120]
000066ca  e821020000        call    std::vector<GfDatHead_t,...tor<GfDatHead_t> >::_M_default_append          [2]
000066cf  ebb2              jmp     0x6683
```

Crash Information

```
---CRASH SUMMARY---
Filename: 0/crashes.2021-07-08-06:48:19/id:000003,sig:06,src:000002,time:7983168,op:flip1,pos:35.gf
SHA1: d1e3930a21198a5f47b4a74172e3b521d24b5404
Classification: EXPLOITABLE
Hash: d534e5b0051653dc75a9212440169e08.740bedddcd989b50806d5aa54a764319
Command: ../AnyCubicPluginLinux 0/crashes.2021-07-08-06:48:19/id:000003,sig:06,src:000002,time:7983168,op:flip1,pos:35.gf out.pwx
Faulting Frame:
   operator new(unsigned long) @ 0x00007ffff7e78b39: in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28
Disassembly:
   0x00007ffff7abb8ba: xor edx,edx
   0x00007ffff7abb8bc: mov rsi,r9
   0x00007ffff7abb8bf: mov edi,0x2
   0x00007ffff7abb8c4: mov eax,0xe
   0x00007ffff7abb8c9: syscall
=> 0x00007ffff7abb8cb: mov rax,QWORD PTR [rsp+0x108]
   0x00007ffff7abb8d3: sub rax,QWORD PTR fs:0x28
   0x00007ffff7abb8dc: jne 0x7ffff7abb904 <__GI_raise+260>
   0x00007ffff7abb8de: mov eax,r8d
   0x00007ffff7abb8e1: add rsp,0x118
Stack Head (12 entries):
   __GI_raise                @ 0x00007ffff7abb8cb: in (BL)
   __GI_abort                @ 0x00007ffff7aa0864: in (BL)
   __libc_message            @ 0x00007ffff7b03af6: in (BL)
   malloc_printerr           @ 0x00007ffff7b0c46c: in (BL)
   _int_malloc               @ 0x00007ffff7b0fb14: in (BL)
   __GI___libc_malloc        @ 0x00007ffff7b115d4: in (BL)
   operator                  @ 0x00007ffff7e78b39: in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28
   std::vector<PhotonsLayerH @ 0x0000555555558a01: in /home/fuzz/Desktop/chitubox/resource/plugin/AnycubicPlugin/AnyCubicPluginLinux
   PwsFile::WritePrepare(GfF @ 0x00005555555579df: in /home/fuzz/Desktop/chitubox/resource/plugin/AnycubicPlugin/AnyCubicPluginLinux
   PwsFile::ZipImages(GfFile @ 0x00005555555588d6: in /home/fuzz/Desktop/chitubox/resource/plugin/AnycubicPlugin/AnyCubicPluginLinux
   Encode2GfFile(char        @ 0x00005555555569a0: in /home/fuzz/Desktop/chitubox/resource/plugin/AnycubicPlugin/AnyCubicPluginLinux
   main                      @ 0x0000555555556619: in /home/fuzz/Desktop/chitubox/resource/plugin/AnycubicPlugin/AnyCubicPluginLinux
Registers:
rax=0x0000000000000000 rbx=0x00007ffff7a78f80 rcx=0x00007ffff7abb8cb rdx=0x0000000000000000
rsi=0x00007fffffffd560 rdi=0x0000000000000002 rbp=0x00007fffffffd8b0 rsp=0x00007fffffffd560
 r8=0x0000000000000000  r9=0x00007fffffffd560 r10=0x0000000000000008 r11=0x0000000000000246
r12=0x00007fffffffd7d0 r13=0x0000000000000010 r14=0x00007ffff7fc7000 r15=0x0000000000000001
rip=0x00007ffff7abb8cb efl=0x0000000000000246  cs=0x0000000000000033  ss=0x000000000000002b
 ds=0x0000000000000000  es=0x0000000000000000  fs=0x0000000000000000  gs=0x0000000000000000
Extra Data:
   Description: Heap error
   Short description: HeapError (10/22)
   Explanation: The target's backtrace indicates that libc has detected a heap error or that the target was executing a heap function when
it stopped.
---END SUMMARY---
```

Timeline

2021-09-28 - Vendor Disclosure
2021-10-29- 30 day follow up
2021-11-18 - 45+ day follow up
2021-12-13 - Final follow up
2022-01-10 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1372

NEXT REPORT TALOS-2021-1387
