Talos Vulnerability Report

TALOS-2020-1125

Pixar OpenUSD binary file format specs memory corruption

NOVEMBER 12, 2020

CVE NUMBER

CVE-2020-13524

Summary

An out-of-bounds memory corruption vulnerability exists in the way Pixar OpenUSD 20.05 uses SPECS data from binary USD files. A specially crafted malformed file can trigger an out-of-bounds memory access and modification which results in memory corruption. To trigger this vulnerability, the victim needs to access an attacker-provided malformed file.

Tested Versions

Pixar OpenUSD 20.05
Apple macOS Catalina 10.15.3

Product URLs

https://openusd.org

CVSSv3 Score

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

CWE

CWE-119 - Improper Restriction of Operations within the Bounds of a Memory Buffer

Details

OpenUSD stands for Open Universal Scene Descriptor and is a software suite by Pixar that facilitates, among other things, interchange of arbitrary 3-D scenes that may be composed of many elemental assets.

Most notably, USD and its backing file format `usd` are used on Apple iOS and macOS as part of ModelIO framework in support of SceneKit and ARKit for sharing and displaying 3D scenes in, for example, augmented reality applications. On macOS, these files are automatically rendered to generate thumbnails, while on iOS they can be shared via iMessage and opened with user interaction.

USD binary file format consists of a header pointing to a table of contents that in turn points to individual sections that comprise the whole file. The `SPECS` section of the file consists of three distinct arrays of integers which are used to connect previously reconstructed paths with sets of fields and their types. For a sample file, these might look like:

```
Path Indexes: [0, 1, 2, 3]
FSet Indexes: [0, 5, 9, 12]
Spec Types: [7, 6, 6, 6]
```

Path indices point into PATHS array, fset indices point into start of field sets and specs into tokens. Specs structure can be considered a root structure for the given USD scene as everything flows from it. When the memory structures are reconstructed by parsing the file, these are being additionally processed into object model after the file is fully parsed. For example, the following code from `crateData.cpp`:

```
    struct _SpecToPair {
        using result_type = _FlatMap::value_type;
        explicit _SpecToPair(CrateFile *crateFile) : crateFile(crateFile) {}
        result_type operator()(CrateFile::Spec const &spec) const {
                result_type r(crateFile->GetPath(spec.pathIndex),                                    [1]
                              _FlatSpecData(Usd_EmptySharedTag));
            TF_AXIOM(!r.first.IsTargetPath());
            return r;
        }
        CrateFile *crateFile;
    };
```

Note that at [1] in the above code a property `pathIndex` is directly dereferenced and a concrete path object is retrieved from path vector via `GetPath`. Value passed into `GetPath` is used as an index into a vector of previously reconstructed paths. Since no check is performed to ensure the index is less than the size of the vector, out of bounds memory access can happen. By supplying a large value for path index inside SPECS structure in the binary file format, an out of bounds memory access will lead to undefined behavior. These indices are 32 bit integers allowing for a large range of out of bounds access values in the above code at [1] and other places inside `_PopulateFromCrateFile` function. With precise memory layout control and in combination with other reported issues, it is possible that this vulnerability could be abused to achieve arbitrary code execution.

Crash Information

Different crashes can be observed, depending on the memory layout and out of bounds value accessed. Following example shows a wild pointer:

```
2020-07-10 10:52:52.988754-0500 qlmanage[30695:1555246] [General] Disabling connection to window server
2020-07-10 10:52:52.989719-0500 qlmanage[30695:1555246] [General] Number of mach ports: 43
2020-07-10 10:52:52.991323-0500 qlmanage[30695:1555266] [General] Cache is disabled (real server: NO - cache enabled: NO)
2020-07-10 10:52:53.042763-0500 qlmanage[30695:1555262] MessageTracer: Falling back to default whitelist
-------------------------------------------------------------------------------------------------------------------------------[regs]
RAX: 0x5000000010062F2F  RBX: 0x0000000104038600  RBP: 0x000070000A0E3220  RSP: 0x000070000A0E3220  o d I t s z a p c
RDI: 0x5000000010062F2F  RSI: 0x00000001006321F0  RDX: 0x0000000010061AE00  RCX: 0x0000000000000000  RIP: 0x00007FFF4AD0E28C
R8:  0x0000000000000000  R9:  0x0000000000000001  R10: 0x0000000000000001  R11: 0x0000000000000001  R12: 0x00000001006321D8
R13: 0x000000010064193C  R14: 0x00000001006321F0  R15: 0x0000000010061AEC0
CS:  002B  FS: 0000  GS: 0000
-------------------------------------------------------------------------------------------------------------------------------[code]
@ /System/Library/Frameworks/ModelIO.framework/Versions/A/ModelIO:
->  0x7fff4ad0e28c (0x48028c): 0f b6 47 0e                    movzx   eax, byte ptr [rdi + 0xe]
0x7fff4ad0e290 (0x480290): 5d                           pop     rbp
0x7fff4ad0e291 (0x480291): c3                           ret
0x7fff4ad0e292 (0x480292): 66 2e 0f 1f 84 00 00 00 00 00  nop     word ptr cs:[rax + rax]
0x7fff4ad0e29c (0x48029c): 0f 1f 40 00                  nop     dword ptr [rax]
0x7fff4ad0e2a0 (0x4802a0): 55                           push    rbp
0x7fff4ad0e2a1 (0x4802a1): 48 89 e5                     mov     rbp, rsp
0x7fff4ad0e2a4 (0x4802a4): 48 81 ec 90 00 00 00         sub     rsp, 0x90
-------------------------------------------------------------------------------------------------------------------------------

Process 30695 stopped
* thread #2, queue = 'quicklookd.serialgenerator', stop reason = EXC_BAD_ACCESS (code=EXC_I386_GPFLT)
frame #0: 0x00007fff4ad0e28c ModelIO`___lldb_unnamed_symbol37504$$ModelIO + 12
Target 0: (qlmanage) stopped.
(lldbinit)
```

Same PoC file with address sanitizer produces :

```
=================================================================
==31217==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000059554 at pc 0x7f8a7f0a8847 bp 0x7ffed56e11a0 sp 0x7ffed56e1190
READ of size 4 at 0x603000059554 thread T0
#0 0x7f8a7f0a8846 in
pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_Pa
thPropTag, 24u, 8u, 16384u>::Handle, false, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const>::get() const pxr/usd/sdf/path.h:148
#1 0x7f8a7f0a8846 in pxrInternal_v0_20__pxrReserved__::SdfPath::IsTargetPath() const pxr/usd/sdf/path.cpp:266
#2 0x7f8a745d0a64 in __gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >
std::__find_if<__gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}> >
(__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>, std::random_access_iterator_tag)
pxr/usd/usd/crateData.cpp:774
#3 0x7f8a74619e5d in __gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >
std::__find_if<__gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}> >
(__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>) /usr/include/c++/9/bits/stl_algo.h:161
#4 0x7f8a74619e5d in __gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >
std::__remove_if<__gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}> >
(__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>,
__gnu_cxx::__ops::_Iter_pred<pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>) /usr/include/c++/9/bits/stl_algo.h:863
#5 0x7f8a74619e5d in __gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >
std::remove_if<__gnu_cxx::__normal_iterator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec*,
std::vector<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec,
std::allocator<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec> > >,
pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}>
(pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1},
pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1},
pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile()::
{lambda(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Spec const&)#1}) /usr/include/c++/9/bits/stl_algo.h:939
#6 0x7f8a74619e5d in pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::_PopulateFromCrateFile() pxr/usd/usd/crateData.cpp:775
#7 0x7f8a745708b7 in pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::Open(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) pxr/usd/usd/crateData.cpp:200
#8 0x7f8a745708b7 in pxrInternal_v0_20__pxrReserved__::Usd_CrateData::Open(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) pxr/usd/usd/crateData.cpp:1205
#9 0x7f8a7625247b in pxrInternal_v0_20__pxrReserved__::UsdUsdcFileFormat::Read(pxrInternal_v0_20__pxrReserved__::SdfLayer*,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, bool) const pxr/usd/usd/usdcFileFormat.cpp:95
#10 0x7f8a7ec73f4b in pxrInternal_v0_20__pxrReserved__::SdfLayer::_Read(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, bool)
pxr/usd/sdf/layer.cpp:1045
#11 0x7f8a7ed3e9ed in pxrInternal_v0_20__pxrReserved__::TfRefPtr<pxrInternal_v0_20__pxrReserved__::SdfLayer>
pxrInternal_v0_20__pxrReserved__::SdfLayer::_OpenLayerAndUnlockRegistry<tbb::queuing_rw_mutex::scoped_lock>
(tbb::queuing_rw_mutex::scoped_lock&, pxrInternal_v0_20__pxrReserved__::SdfLayer::_FindOrOpenLayerInfo const&, bool)
pxr/usd/sdf/layer.cpp:3072
#12 0x7f8a7ed0fa4f in pxrInternal_v0_20__pxrReserved__::SdfLayer::FindOrOpen(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, std::map<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >,
std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::less<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >, std::allocator<std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > > const&)
pxr/usd/sdf/layer.cpp:819
#13 0x55f659e00baa in main pxr/usd/bin/sdfdump/sdfdump.cpp:522
#14 0x7f8a7c66b0b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
#15 0x55f659e0b6bd in _start (build/bin/sdfdump+0x2a6bd)

0x603000059554 is located 4 bytes to the right of 32-byte region [0x603000059530,0x603000059550)
allocated by thread T0 here:
#0 0x7f8a7f905947 in operator new(unsigned long) (/lib/x86_64-linux-gnu/libasan.so.5+0x10f947)
#1 0x7f8a7f111c7f in __gnu_cxx::new_allocator<pxrInternal_v0_20__pxrReserved__::SdfPath>::allocate(unsigned long, void const*)
/usr/include/c++/9/ext/new_allocator.h:114
#2 0x7f8a7f111c7f in std::allocator_traits<std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath>
>::allocate(std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath>&, unsigned long) /usr/include/c++/9/bits/alloc_traits.h:444
#3 0x7f8a7f111c7f in std::_Vector_base<pxrInternal_v0_20__pxrReserved__::SdfPath, std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath>
>::_M_allocate(unsigned long) /usr/include/c++/9/bits/stl_vector.h:343
#4 0x7f8a7f111c7f in std::vector<pxrInternal_v0_20__pxrReserved__::SdfPath, std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath>
>::_M_default_append(unsigned long) /usr/include/c++/9/bits/vector.tcc:635

SUMMARY: AddressSanitizer: heap-buffer-overflow pxr/usd/sdf/path.h:148 in
pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_Pa
thPropTag, 24u, 8u, 16384u>::Handle, false, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const>::get() const
Shadow bytes around the buggy address:
0x0c0680003250: fa fa fd fd fd fd fa fa fd fd fd fd fa fa 00 00
0x0c0680003260: 00 00 fa fa 00 00 00 fa fa fa 00 00 00 00 fa fa
0x0c0680003270: fd fd fd fd fd fa fa 00 00 00 05 fa fa fd fd fd
0x0c0680003280: fa fa fd fd fd fd fa fa 00 00 00 00 fa fa fd fd
0x0c0680003290: fd fd fa fa fd fd fd fa fa fa fd fd fd fa fa fa
=>0x0c06800032a0: 00 00 00 fa fa fa 00 00 00 00[fa]fa fd fd fd fa
0x0c06800032b0: fa fa fd fd fd fd fa fa fd fd fd fa fa fa fa fa
0x0c06800032c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c06800032d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c06800032e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c06800032f0: 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
Freed heap region:       fd
Stack left redzone:      f1
Stack mid redzone:       f2
Stack right redzone:     f3
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
Left alloca redzone:     ca
Right alloca redzone:    cb
Shadow gap:              cc
==31217==ABORTING
```

Timeline

2020-07-17 - Vendor Disclosure
2020-11-12 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-1094

NEXT REPORT TALOS-2020-1145