Talos Vulnerability Report

TALOS-2020-1120

Pixar OpenUSD Binary File Format Decompressed Path Rebuilding Memory corruption

NOVEMBER 12, 2020

CVE NUMBER

CVE-2020-13520

Summary

An out of bounds memory corruption vulnerability exists in the way Pixar OpenUSD 20.05 reconstructs paths from binary USD files. A specially crafted malformed file can trigger an out of bounds memory modification which can result in remote code execution. To trigger this vulnerability, 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

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

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 `PATHS` section of the file consists of three distinct arrays of integers which are used to reconstruct the SDF paths during file parsing. These arrays represent indices and are: `pathIndexes`, `elementTokenIndexes` and `jumps`. After decoding these three arrays, the actual paths need to be reconstructed and this is done by invoking the following code:

```
void
CrateFile::_BuildDecompressedPathsImpl(
vector<uint32_t> const &pathIndexes,
vector<int32_t> const &elementTokenIndexes,
vector<int32_t> const &jumps,
size_t curIndex,
SdfPath parentPath,
WorkArenaDispatcher &dispatcher)
{
bool hasChild = false, hasSibling = false;
do {
    auto thisIndex = curIndex++;
    if (parentPath.IsEmpty()) {
        parentPath = SdfPath::AbsoluteRootPath();
        _paths[pathIndexes[thisIndex]] = parentPath;           [1]
    } else {
        int32_t tokenIndex = elementTokenIndexes[thisIndex];
        bool isPrimPropertyPath = tokenIndex < 0;
        tokenIndex = std::abs(tokenIndex);
        auto const &elemToken = _tokens[tokenIndex];
        _paths[pathIndexes[thisIndex]] =                       [2]
            isPrimPropertyPath ?
            parentPath.AppendProperty(elemToken) :
            parentPath.AppendElementToken(elemToken);
    }

    // If we have either a child or a sibling but not both, then just
    // continue to the neighbor.  If we have both then spawn a task for the
    // sibling and do the child ourself.  We think that our path trees tend
    // to be broader more often than deep.

    hasChild = (jumps[thisIndex] > 0) || (jumps[thisIndex] == -1);
    hasSibling = (jumps[thisIndex] >= 0);
```

Inputs to the above method are three arrays of indices. To illustrate the algorithm, we can observe the following contents from a regular USD file:

```
Path Indexes: [0, 1, 2, 3]
Element Token Indexes: [7, 5, 12, 11]
Jumps Indexes: [-1, -1, 0, -2]
```

Path indices are incremental, element token indices refer to string tokens in TOKENS section and jumps encode whether child or sibling elements for a path exist. For the concrete example, the above arrays might encode following paths:

```
</> : SdfSpecTypePseudoRoot
</World> : SdfSpecTypePrim
</World/camera> : SdfSpecTypePrim
</World/scope> : SdfSpecTypePrim
```

When parsing the USDC file, these are reconstructed by calling `_BuildDecompressedPathsImpl` method. The vulnerability arises from the fact that path indices aren't checked to ensure they fall inside the appropriate vector. The indices in `pathIndexes` are 32 bit integers allowing for a large range of out of bounds access values in the above code at [1] and [2]. In both instances an object of `SdfPath` type will be written to the supplied address. A malformed USDC file with very large `pathIndexes` values can cause multiple out of bounds memory writes which can lead to further memory corruption and potentially lead to arbitrary code execution.

Crash Information

The attached proof of concept trigger crashes with the following ASAN message:

```
AddressSanitizer:DEADLYSIGNAL
=================================================================
==125593==ERROR: AddressSanitizer: SEGV on unknown address 0x6032aab040c8 (pc 0x7fbec5883737 bp 0x0fff938127f6 sp 0x7ffc9c093fa0 T0)
==125593==The signal is caused by a READ memory access.
    #0 0x7fbec5883736 in
pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_Pa
thPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const>::operator==
(pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_P
athPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const> const&) const ./USD-
20.05/pxr/usd/sdf/path.h:170
    #1 0x7fbec5883736 in
pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_Pa
thPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const>::operator=
(pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_P
athPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const> const&) ./USD-20.05/pxr/usd/sdf/path.h:118
    #2 0x7fbebb380f44 in pxrInternal_v0_20__pxrReserved__::SdfPath::operator=(pxrInternal_v0_20__pxrReserved__::SdfPath const&) ./USD-
20.05/pxr/usd/sdf/path.h:288
    #3 0x7fbebb380f44 in pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_BuildDecompressedPathsImpl(std::vector<unsigned int,
std::allocator<unsigned int> > const&, std::vector<int, std::allocator<int> > const&, std::vector<int, std::allocator<int> > const&,
unsigned long, pxrInternal_v0_20__pxrReserved__::SdfPath, pxrInternal_v0_20__pxrReserved__::WorkArenaDispatcher&) ./USD-
20.05/pxr/usd/usd/crateFile.cpp:3461
    #4 0x7fbebb55223a in void
pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_ReadCompressedPaths<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile:
:_Reader<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMappi
ng*> > >
(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_Reader<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInterna
l_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMapping*> >, pxrInternal_v0_20__pxrReserved__::WorkArenaDispatcher&) ./USD-
20.05/pxr/usd/usd/crateFile.cpp:3441
    #5 0x7fbebb628bc8 in void
pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_ReadPaths<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_Reader<p
xrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMapping*> > >
(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_Reader<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInterna
l_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMapping*> >) ./USD-20.05/pxr/usd/usd/crateFile.cpp:3337
    #6 0x7fbebb6d7a1d in void
pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_ReadStructuralSections<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFi
le::_Reader<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMa
pping*> > >
(pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_Reader<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::_MmapStream<pxrInterna
l_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMapping*> >, long) ./USD-20.05/pxr/usd/usd/crateFile.cpp:3038
    #7 0x7fbebb3df25b in pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_InitMMap() ./USD-20.05/pxr/usd/usd/crateFile.cpp:2140
    #8 0x7fbebb3e7a3a in pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::CrateFile(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&, boost::intrusive_ptr<pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::_FileMapping>,
std::shared_ptr<pxrInternal_v0_20__pxrReserved__::ArAsset> const&) ./USD-20.05/pxr/usd/usd/crateFile.cpp:2104
    #9 0x7fbebb3ebdf6 in pxrInternal_v0_20__pxrReserved__::Usd_CrateFile::CrateFile::Open(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&) ./USD-20.05/pxr/usd/usd/crateFile.cpp:2051
    #10 0x7fbebb27081b in pxrInternal_v0_20__pxrReserved__::Usd_CrateDataImpl::Open(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) ./USD-20.05/pxr/usd/usd/crateData.cpp:198
    #11 0x7fbebb27081b in pxrInternal_v0_20__pxrReserved__::Usd_CrateData::Open(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&) ./USD-20.05/pxr/usd/usd/crateData.cpp:1205
    #12 0x7fbebcf5247b 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 ./USD-
20.05/pxr/usd/usd/usdcFileFormat.cpp:95
    #13 0x7fbec5934f4b 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) ./USD-
20.05/pxr/usd/sdf/layer.cpp:1045
    #14 0x7fbec59ff9ed 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) ./USD-
20.05/pxr/usd/sdf/layer.cpp:3072
    #15 0x7fbec59d0a4f 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&) ./USD-
20.05/pxr/usd/sdf/layer.cpp:819
    #16 0x558a1aabfbaa in main ./USD-20.05/pxr/usd/bin/sdfdump/sdfdump.cpp:522
    #17 0x7fbec332c0b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
    #18 0x558a1aaca6bd in _start (./USD-20.05/build/bin/sdfdump+0x2a6bd)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV ./USD-20.05/pxr/usd/sdf/path.h:170 in
pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_Pa
thPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const>::operator==
(pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_P
athPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const> const&) const
==125593==ABORTING
```

Timeline

2020-07-07 - Vendor Disclosure

2020-11-12 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.