Talos Vulnerability Report

TALOS-2020-1226

3MF Consortium lib3mf NMR::COpcPackageReader::releaseZIP() use-after-free vulnerability

MARCH 10, 2021

CVE NUMBER

CVE-2021-21772

Summary

A use-after-free vulnerability exists in the NMR::COpcPackageReader::releaseZIP() functionality of 3MF Consortium lib3mf 2.0.0. A specially crafted 3MF file can lead to code execution. An attacker can provide a malicious file to trigger this vulnerability.

Tested Versions

3MF Consortium lib3mf 2.0.0

Product URLs

https://github.com/3MFConsortium/lib3mf

CVSSv3 Score

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

CWE

CWE-416 - Use After Free

Details

The lib3mf library is an open-source implementation of the 3MF file format and standard, mainly used for 3D-printing. Lib3mf provides input, output and conversion capabilities, along with API bindings for various languages (C, C++, C#, Go, NodeJS, Pascal and Python). The 3MF standard has been adopted in a variety of products and lib3mf itself has been confirmed to be used in open-source programs like OpenSCAD and LibCGAL.

When reading in a .3mf file (essentially a .zip file with a particular layout inside), via lib3mf, one needs to call one of the aptly named `Lib3MF::CReader::ReadFromBuffer` or `Lib3MF::CReader::ReadFromFile` functions, depending on the need of the programmer. Aside from input source, they both do the same thing:

```
void CReader::ReadFromBuffer (const Lib3MF_uint64 nBufferBufferSize, const Lib3MF_uint8 * pBufferBuffer)
{
    NMR::PImportStream pImportStream = std::make_shared<NMR::CImportStream_Shared_Memory>(pBufferBuffer, nBufferBufferSize); // [1]

    try {
        reader().readStream(pImportStream); // [2]
    }
    catch (NMR::CNMRException&e) {
        if (e.getErrorCode() == NMR_USERABORTED) {
            throw ELib3MFInterfaceException(LIB3MF_ERROR_CALCULATIONABORTED);
        }
        else throw e;
    }
}
```

[1] is a stream object initialization/allocation, and then we read the stream at [2]. Proceeding into `CModelReader_3MF::readStream`:

```
    void CModelReader_3MF::readStream(_In_ PImportStream pStream)
    {
        __NMRASSERT(pStream != nullptr);

        nfBool bHasModel = false;

        m_pProgressMonitor->SetProgressIdentifier(ProgressIdentifier::PROGRESS_READSTREAM);

        m_pProgressMonitor->SetProgressIdentifier(ProgressIdentifier::PROGRESS_EXTRACTOPCPACKAGE);

        // Extract Stream from Package
        PImportStream pModelStream = extract3MFOPCPackage(pStream); // [1]
```

Nothing fancy yet, [1] just attempts to extract a 3mf OPC package from the input stream. Without getting to in the weeds with the code path, eventually `extract3MFOPCPackage()` leads us to the creation of a `COpcPackageReader` object:

```
    COpcPackageReader::COpcPackageReader(_In_ PImportStream pImportStream, _In_ PModelReaderWarnings pWarnings, _In_ PProgressMonitor
pProgressMonitor)
        : m_pWarnings(pWarnings), m_pProgressMonitor(pProgressMonitor)
    {
        if (!pImportStream)
            throw CNMRException(NMR_ERROR_INVALIDPARAM);

        if (!pProgressMonitor)
            throw CNMRException(NMR_ERROR_INVALIDPARAM);

        m_ZIPError.str = nullptr;
        m_ZIPError.sys_err = 0;
        m_ZIPError.zip_err = 0;
        m_ZIParchive = nullptr;
        m_ZIPsource = nullptr;

        try {
            // determine stream size
            nfUint64 nStreamSize = pImportStream->retrieveSize();
            pImportStream->seekPosition(0, true);

            if (nStreamSize == 0)
                throw CNMRException(NMR_ERROR_COULDNOTGETSTREAMPOSITION);

            // create ZIP objects
            zip_error_init(&m_ZIPError);

            bool bUseCallback = true;
            if (bUseCallback) {
                // read ZIP from callback: faster and requires less memory
                m_ZIPsource = zip_source_function_create(custom_zip_source_callback, pImportStream.get(), &m_ZIPError); //[1]
            }
     //[...]
```

This is where the non-wrapper code starts to operate, our `COpcPackageReader` object is just a wrapper around https://libzip.org/, using the libzip api to deal with the bulk of the work since `.3mf` files are just fancy .zips. The `zip_source_function_create` function at [1] is where the libzip structs are allocated, as it calls `_zip_source_new`:

```
zip_source_t * _zip_source_new(zip_error_t *error) {
    zip_source_t *src;

    if ((src = (zip_source_t *)malloc(sizeof(*src))) == NULL) { // [1]
    zip_error_set(error, ZIP_ER_MEMORY, 0);
    return NULL;
    }

src->src = NULL;
src->cb.f = NULL;
src->ud = NULL;
src->open_count = 0;
src->write_state = ZIP_SOURCE_WRITE_CLOSED;
src->source_closed = false;
src->source_archive = NULL;
src->refcount = 1;
zip_error_init(&src->error);
src->eof = false;
src->had_read_error = false;

return src; }
```

We only really care about the `zip_source_t` object allcoated at [1], as we'll see soon. Stepping back up to the `COpcPackageReader` object initialization, the allocated `zip_source_t` object is returned back into the `m_ZIPsource` variable and we proceed to read the file:

```
   try {
        // [...]

        bool bUseCallback = true;
        if (bUseCallback) {
            // read ZIP from callback: faster and requires less memory
            m_ZIPsource = zip_source_function_create(custom_zip_source_callback, pImportStream.get(), &m_ZIPError);  //[1]
        }
        // [...]

        if (m_ZIPsource == nullptr)
            throw CNMRException(NMR_ERROR_COULDNOTREADZIPFILE);

        m_ZIParchive = zip_open_from_source(m_ZIPsource, ZIP_RDONLY | ZIP_CHECKCONS, &m_ZIPError);
        if (m_ZIParchive == nullptr) {
            m_ZIParchive = zip_open_from_source(m_ZIPsource, ZIP_RDONLY, &m_ZIPError);
            if (m_ZIParchive == nullptr)
                throw CNMRException(NMR_ERROR_COULDNOTREADZIPFILE);
            else
                m_pWarnings->addException(CNMRException(NMR_ERROR_ZIPCONTAINSINCONSISTENCIES), mrwInvalidMandatoryValue);
        }

        // Get ZIP Content
        nfInt64 nEntryCount = zip_get_num_entries(m_ZIParchive, ZIP_FL_UNCHANGED);
        if (nEntryCount < 0)
            throw CNMRException(NMR_ERROR_COULDNOTREADZIPFILE);

        // List and stat Entries
        nfUint64 nUnzippedFileSize = 0;
        for (nfInt64 nIndex = 0; nIndex < nEntryCount; nIndex++) {
            const char * pszName = zip_get_name(m_ZIParchive, (nfUint64) nIndex, ZIP_FL_ENC_GUESS);
            m_ZIPEntries.insert(std::make_pair(pszName, nIndex));

            zip_stat_t Stat;
            nfInt32 nResult = zip_stat_index(m_ZIParchive, nIndex, ZIP_FL_UNCHANGED, &Stat);
            if (nResult != 0)
                throw CNMRException(NMR_ERROR_COULDNOTSTATZIPENTRY);

            nUnzippedFileSize += Stat.size;
        }

        m_pProgressMonitor->SetMaxProgress(double(nUnzippedFileSize));
        m_pProgressMonitor->ReportProgressAndQueryCancelled(true);

        readContentTypes();
        readRootRelationships();
    }

    }
    catch (...)
    {
        releaseZIP(); //[2]
        throw;
    }
```

It's not too important to actually read the code above, just worth noting alot of processing happens on the input .zip file from our `zip_source_t` object at [1], and if any exception are raised, we hit `releaseZIP()` at [2]. To proceed into this call path:

```
void COpcPackageReader::releaseZIP()
{
    if (m_ZIParchive != nullptr)
        zip_close(m_ZIParchive);  // [1]

    if (m_ZIPsource != nullptr)
        zip_source_close(m_ZIPsource); // [2]

    zip_error_fini(&m_ZIPError);
    m_Buffer.resize(0);

    m_ZIPsource = nullptr;
    m_ZIParchive = nullptr;
}
```

To quickly describe what `zip_close` [1] does, the libzip documentation: `The zip_close() function writes any changes made to archive to disk. If archive contains no files, the file is completely removed (no empty archive is written). If successful, archive is freed. Otherwise archive is left unchanged and must still be freed.` As for the actual implimentation in `libzip-1.7.3/lib/zip_close.c`:

```
ZIP_EXTERN int
zip_close(zip_t *za) {
    zip_uint64_t i, j, survivors, unchanged_offset;
    zip_int64_t off;
    int error;
    zip_filelist_t *filelist;
    int changed;

    if (za == NULL)
    return -1;

    changed = _zip_changed(za, &survivors);

    /* don't create zip files with no entries */
    if (survivors == 0) {
        if ((za->open_flags & ZIP_TRUNCATE) || changed) {
            if (zip_source_remove(za->src) < 0) {
            if (!((zip_error_code_zip(zip_source_error(za->src)) == ZIP_ER_REMOVE) && (zip_error_code_system(zip_source_error(za->src)) ==
ENOENT))) {
                _zip_error_set_from_source(&za->error, za->src);
                return -1;
            }
            }
        }
        zip_discard(za);
        return 0;
    }

    if (!changed) {
        zip_discard(za); //[1]
        return 0;  // The zip_discard() function closes archive and frees the memory allocated for it. Any changes to the archive are not
written to disk and discarded.
    }
    //[...] // if (changed)
```

Assuming no changes are made to the input zip file, we hit the code path at [1], going into `zip_discard`:

```
void
zip_discard(zip_t *za) {
    zip_uint64_t i;

    if (za == NULL)
    return;

    if (za->src) {
        zip_source_close(za->src);
        zip_source_free(za->src); // [1]
    }
//[...]
```

At [1] above, we reach the `zip_source_free` function, which critically frees the initially allocated `zip_source_t` struct (from the `m_ZIPsource = zip_source_function_create(custom_zip_source_callback, pImportStream.get(), &m_ZIPError);` line) inside of our `COpcPackageReader` object at [2] below:

```
ZIP_EXTERN void
zip_source_free(zip_source_t *src) {
    if (src == NULL)
    return;

    if (src->refcount > 0) {
    src->refcount--;
    }
    if (src->refcount > 0) {
    return;
    }

    if (ZIP_SOURCE_IS_OPEN_READING(src)) {
    src->open_count = 1; /* force close */
    zip_source_close(src);
    }
    if (ZIP_SOURCE_IS_OPEN_WRITING(src)) {
    zip_source_rollback_write(src);
    }

    if (src->source_archive && !src->source_closed) {
    _zip_deregister_source(src->source_archive, src);
    }

    (void)_zip_source_call(src, NULL, 0, ZIP_SOURCE_FREE);

    if (src->src) {
    zip_source_free(src->src);
    }

    free(src); // [2]
}
```

After this chain of events and we step back up to `COpcPackageReader::releaseZip()`:

```
void COpcPackageReader::releaseZIP()
{
    if (m_ZIParchive != nullptr)
        zip_close(m_ZIParchive);  // [1]

    if (m_ZIPsource != nullptr)
        zip_source_close(m_ZIPsource); // [2]

    zip_error_fini(&m_ZIPError);
    m_Buffer.resize(0);

    m_ZIPsource = nullptr;
    m_ZIParchive = nullptr;
}
```

For the astute reader, the issue might be clear already. At [1], we eventually free the `m_ZIParchive->src` struct, which is also known as the `m_ZIPsource` object that is subsequently closed at [2]. Thus, the `zip_source_close` function can be passed an arbitrary user-controlled object, assuming that an attacker can win a race and allocate an object into the appropriate slot in between the `free(m_ZIPsource)` inside [1] and the use at [2], resulting in a classic use-after-free vulnerability inside `zip_source_close`:

```
int
zip_source_close(zip_source_t *src) { // src object can be user-controlled.
    if (!ZIP_SOURCE_IS_OPEN_READING(src)) {
        zip_error_set(&src->error, ZIP_ER_INVAL, 0);
        return -1;
    }

    src->open_count--;
    if (src->open_count == 0) {
    _zip_source_call(src, NULL, 0, ZIP_SOURCE_CLOSE);

    if (ZIP_SOURCE_IS_LAYERED(src)) {
        if (zip_source_close(src->src) < 0) {
        zip_error_set(&src->error, ZIP_ER_INTERNAL, 0);
        }
    }
    }

    return 0;
}
```

Crash Information

```
==4692==ERROR: AddressSanitizer: heap-use-after-free on address 0x607000000050 at pc 0x7f2343babb3b bp 0x7ffd6b561a70 sp 0x7ffd6b561a68
READ of size 4 at 0x607000000050 thread T0
    #0 0x7f2343babb3a in zip_source_close //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_source_close.c:40:10
    #1 0x7f23442242e7 in NMR::COpcPackageReader::releaseZIP() //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Common/OPC/NMR_OpcPackageReader.cpp:219:4
    #2 0x7f234422251b in NMR::COpcPackageReader::COpcPackageReader(std::shared_ptr<NMR::CImportStream>,
std::shared_ptr<NMR::CModelReaderWarnings>, std::shared_ptr<NMR::CProgressMonitor>) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Common/OPC/NMR_OpcP
ackageReader.cpp:179:4
    #3 0x7f234427af0c in void __gnu_cxx::new_allocator<NMR::COpcPackageReader>::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(NMR::COpcPackageReader*,
 std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/ext/new_allocator.h:147:23
    #4 0x7f234427a7d9 in void std::allocator_traits<std::allocator<NMR::COpcPackageReader> >::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>(std::alloc
ator<NMR::COpcPackageReader>&, NMR::COpcPackageReader*, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/alloc_traits
.h:484:8
    #5 0x7f234427a016 in std::_Sp_counted_ptr_inplace<NMR::COpcPackageReader, std::allocator<NMR::COpcPackageReader>,
(__gnu_cxx::_Lock_policy)2>::_Sp_counted_ptr_inplace<std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::sha
red_ptr<NMR::CProgressMonitor>&>(std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/
bits/shared_ptr_base.h:548:4
    #6 0x7f2344279972 in std::__shared_count<(__gnu_cxx::_Lock_policy)2>::__shared_count<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProg
ressMonitor>&>(NMR::COpcPackageReader*&, std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-li
nux-gnu/9/../../../../include/c++/9/bits/shared_ptr_base.h:680:6
    #7 0x7f23442795e2 in std::__shared_ptr<NMR::COpcPackageReader,
(__gnu_cxx::_Lock_policy)2>::__shared_ptr<std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgress
Monitor>&>(std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/
c++/9/bits/shared_ptr_base.h:1344:14
    #8 0x7f2344279317 in std::shared_ptr<NMR::COpcPackageReader>::shared_ptr<std::allocator<NMR::COpcPackageReader>,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::_Sp_alloc_shared
_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/shared_ptr.h:359:4
    #9 0x7f2344278fb8 in std::shared_ptr<NMR::COpcPackageReader> std::allocate_shared<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgres
sMonitor>&>(std::allocator<NMR::COpcPackageReader> const&, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_pt
r.h:701:14
    #10 0x7f2344273877 in std::shared_ptr<NMR::COpcPackageReader> std::make_shared<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::shared_ptr<NMR::CImportStr
eam>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_ptr.h:717:14
    #11 0x7f234427244f in NMR::CModelReader_3MF_Native::extract3MFOPCPackage(std::shared_ptr<NMR::CImportStream>)
//boop/assorted_fuzzing/lib3mf/lib3mf-2.0.0/Source/Model/Reader/NMR_ModelReader_3MF_Native.cpp:53:22
    #12 0x7f23443cff16 in NMR::CModelReader_3MF::readStream(std::shared_ptr<NMR::CImportStream>) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Model/Reader/NMR_ModelReader_3MF.cpp:130:32
    #13 0x7f234412a872 in Lib3MF::Impl::CReader::ReadFromBuffer(unsigned long, unsigned char const*) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/API/lib3mf_reader.cpp:87:12
    #14 0x7f234454055a in lib3mf_reader_readfrombuffer //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/build/Autogenerated/Source/Implementation/lib3mf_interfacewrapper.cpp:381:13
    #15 0x55247e in Lib3MF::CReader::ReadFromBuffer(Lib3MF::CInputVector<unsigned char> const&) //boop/assorted_fuzzing/lib3mf/./lib3mf-
2.0.0/build/Autogenerated/Bindings/Cpp/lib3mf_implicit.hpp:1622:14
    #16 0x55173d in LLVMFuzzerTestOneInput //boop/assorted_fuzzing/lib3mf/./fuzz_3mf_to_stl.cpp:88:15
    #17 0x45bcf1 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x45bcf1)
    #18 0x447462 in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x447462)
    #19 0x44cf16 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long))
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x44cf16)
    #20 0x475bd2 in main (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x475bd2)
    #21 0x7f23437e80b2 in __libc_start_main /build/glibc-ZN95T4/glibc-2.31/csu/../csu/libc-start.c:308:16
    #22 0x421b2d in _start (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x421b2d)

0x607000000050 is located 48 bytes inside of 80-byte region [0x607000000020,0x607000000070)
freed by thread T0 here:
    #0 0x5215dd in free (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x5215dd)
    #1 0x7f2343bb61da in zip_source_free //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_source_free.c:70:5
    #2 0x7f2343b83bc9 in zip_discard //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_discard.c:53:2
    #3 0x7f2343b6ffe6 in zip_close //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_close.c:79:2
    #4 0x7f2344224255 in NMR::COpcPackageReader::releaseZIP() //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Common/OPC/NMR_OpcPackageReader.cpp:216:4
    #5 0x7f234422251b in NMR::COpcPackageReader::COpcPackageReader(std::shared_ptr<NMR::CImportStream>,
std::shared_ptr<NMR::CModelReaderWarnings>, std::shared_ptr<NMR::CProgressMonitor>) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Common/OPC/NMR_OpcP
ackageReader.cpp:179:4
    #6 0x7f234427af0c in void __gnu_cxx::new_allocator<NMR::COpcPackageReader>::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(NMR::COpcPackageReader*,
 std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/ext/new_allocator.h:147:23
    #7 0x7f234427a7d9 in void std::allocator_traits<std::allocator<NMR::COpcPackageReader> >::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>(std::alloc
ator<NMR::COpcPackageReader>&, NMR::COpcPackageReader*, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/alloc_traits
.h:484:8
    #8 0x7f234427a016 in std::_Sp_counted_ptr_inplace<NMR::COpcPackageReader, std::allocator<NMR::COpcPackageReader>,
(__gnu_cxx::_Lock_policy)2>::_Sp_counted_ptr_inplace<std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::sha
red_ptr<NMR::CProgressMonitor>&>(std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/
bits/shared_ptr_base.h:548:4
    #9 0x7f2344279972 in std::__shared_count<(__gnu_cxx::_Lock_policy)2>::__shared_count<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProg
ressMonitor>&>(NMR::COpcPackageReader*&, std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-li
```

```
nux-gnu/9/../../../../include/c++/9/bits/shared_ptr_base.h:680:6
    #10 0x7f23442795e2 in std::__shared_ptr<NMR::COpcPackageReader,
(__gnu_cxx::_Lock_policy)2>::__shared_ptr<std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgres
sMonitor>&>(std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include
/c++/9/bits/shared_ptr_base.h:1344:14
    #11 0x7f2344279317 in std::shared_ptr<NMR::COpcPackageReader>::shared_ptr<std::allocator<NMR::COpcPackageReader>,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::_Sp_alloc_share
d_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/shared_ptr.h:359:4
    #12 0x7f2344278fb8 in std::shared_ptr<NMR::COpcPackageReader> std::allocate_shared<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgre
ssMonitor>&>(std::allocator<NMR::COpcPackageReader> const&, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_p
tr.h:701:14
    #13 0x7f2344273877 in std::shared_ptr<NMR::COpcPackageReader> std::make_shared<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::shared_ptr<NMR::CImportStr
eam>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_ptr.h:717:14
    #14 0x7f234427244f in NMR::CModelReader_3MF_Native::extract3MFOPCPackage(std::shared_ptr<NMR::CImportStream>)
//boop/assorted_fuzzing/lib3mf/lib3mf-2.0.0/Source/Model/Reader/NMR_ModelReader_3MF_Native.cpp:53:22
    #15 0x7f23443cff16 in NMR::CModelReader_3MF::readStream(std::shared_ptr<NMR::CImportStream>) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Model/Reader/NMR_ModelReader_3MF.cpp:130:32
    #16 0x7f234412a872 in Lib3MF::Impl::CReader::ReadFromBuffer(unsigned long, unsigned char const*) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/API/lib3mf_reader.cpp:87:12
    #17 0x7f234454055a in lib3mf_reader_readfrombuffer //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/build/Autogenerated/Source/Implementation/lib3mf_interfacewrapper.cpp:381:13
    #18 0x55247e in Lib3MF::CReader::ReadFromBuffer(Lib3MF::CInputVector<unsigned char> const&) //boop/assorted_fuzzing/lib3mf/./lib3mf-
2.0.0/build/Autogenerated/Bindings/Cpp/lib3mf_implicit.hpp:1622:14
    #19 0x55173d in LLVMFuzzerTestOneInput //boop/assorted_fuzzing/lib3mf/./fuzz_3mf_to_stl.cpp:88:15
    #20 0x45bcf1 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x45bcf1)
    #21 0x447462 in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x447462)
    #22 0x44cf16 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long))
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x44cf16)
    #23 0x475bd2 in main (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x475bd2)
    #24 0x7f23437e80b2 in __libc_start_main /build/glibc-ZN95T4/glibc-2.31/csu/../csu/libc-start.c:308:16

previously allocated by thread T0 here:
    #0 0x52185d in malloc (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x52185d)
    #1 0x7f2343bb65a8 in _zip_source_new //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_source_function.c:79:32
    #2 0x7f2343bb633f in zip_source_function_create //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_source_function.c:54:15
    #3 0x7f23442216e9 in NMR::COpcPackageReader::COpcPackageReader(std::shared_ptr<NMR::CImportStream>,
std::shared_ptr<NMR::CModelReaderWarnings>, std::shared_ptr<NMR::CProgressMonitor>)
    #4 0x7f234427af0c in void __gnu_cxx::new_allocator<NMR::COpcPackageReader>::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(NMR::COpcPackageReader*,
 std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/ext/new_allocator.h:147:23
    #5 0x7f234427a7d9 in void std::allocator_traits<std::allocator<NMR::COpcPackageReader> >::construct<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>(std::alloc
ator<NMR::COpcPackageReader>&, NMR::COpcPackageReader*, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/alloc_traits
.h:484:8
    #6 0x7f234427a016 in std::_Sp_counted_ptr_inplace<NMR::COpcPackageReader, std::allocator<NMR::COpcPackageReader>,
(__gnu_cxx::_Lock_policy)2>::_Sp_counted_ptr_inplace<std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::sha
red_ptr<NMR::CProgressMonitor>&>(std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/
bits/shared_ptr_base.h:548:4
    #7 0x7f2344279972 in std::__shared_count<(__gnu_cxx::_Lock_policy)2>::__shared_count<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProg
ressMonitor>&>(NMR::COpcPackageReader*&, std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&)
/usr/bin/../lib/gcc/x86_64-li
nux-gnu/9/../../../../include/c++/9/bits/shared_ptr_base.h:680:6
    #8 0x7f23442795e2 in std::__shared_ptr<NMR::COpcPackageReader,
(__gnu_cxx::_Lock_policy)2>::__shared_ptr<std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgress
Monitor>&>(std::_Sp_alloc_shared_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/
c++/9/bits/shared_ptr_base.h:1344:14
    #9 0x7f2344279317 in std::shared_ptr<NMR::COpcPackageReader>::shared_ptr<std::allocator<NMR::COpcPackageReader>,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::_Sp_alloc_shared
_tag<std::allocator<NMR::COpcPackageReader> >, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/bits/shared_ptr.h:359:4
    #10 0x7f2344278fb8 in std::shared_ptr<NMR::COpcPackageReader> std::allocate_shared<NMR::COpcPackageReader,
std::allocator<NMR::COpcPackageReader>, std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&,
std::shared_ptr<NMR::CProgre
ssMonitor>&>(std::allocator<NMR::COpcPackageReader> const&, std::shared_ptr<NMR::CImportStream>&,
std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_p
tr.h:701:14
    #11 0x7f2344273877 in std::shared_ptr<NMR::COpcPackageReader> std::make_shared<NMR::COpcPackageReader,
std::shared_ptr<NMR::CImportStream>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&>
(std::shared_ptr<NMR::CImportStr
eam>&, std::shared_ptr<NMR::CModelReaderWarnings>&, std::shared_ptr<NMR::CProgressMonitor>&) /usr/bin/../lib/gcc/x86_64-linux-
gnu/9/../../../../include/c++/9/bits/shared_ptr.h:717:14
    #12 0x7f234427244f in NMR::CModelReader_3MF_Native::extract3MFOPCPackage(std::shared_ptr<NMR::CImportStream>)
//boop/assorted_fuzzing/lib3mf/lib3mf-2.0.0/Source/Model/Reader/NMR_ModelReader_3MF_Native.cpp:53:22
    #13 0x7f23443cff16 in NMR::CModelReader_3MF::readStream(std::shared_ptr<NMR::CImportStream>) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/Model/Reader/NMR_ModelReader_3MF.cpp:130:32
    #14 0x7f234412a872 in Lib3MF::Impl::CReader::ReadFromBuffer(unsigned long, unsigned char const*) //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/Source/API/lib3mf_reader.cpp:87:12
    #15 0x7f234454055a in lib3mf_reader_readfrombuffer //boop/assorted_fuzzing/lib3mf/lib3mf-
2.0.0/build/Autogenerated/Source/Implementation/lib3mf_interfacewrapper.cpp:381:13
    #16 0x55247e in Lib3MF::CReader::ReadFromBuffer(Lib3MF::CInputVector<unsigned char> const&) //boop/assorted_fuzzing/lib3mf/./lib3mf-
2.0.0/build/Autogenerated/Bindings/Cpp/lib3mf_implicit.hpp:1622:14
    #17 0x55173d in LLVMFuzzerTestOneInput //boop/assorted_fuzzing/lib3mf/./fuzz_3mf_to_stl.cpp:88:15
    #18 0x45bcf1 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x45bcf1)
    #19 0x447462 in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long)
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x447462)
    #20 0x44cf16 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long))
(//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x44cf16)
    #21 0x475bd2 in main (//boop/assorted_fuzzing/lib3mf/3mf_to_stl/3mf_to_stl.bin+0x475bd2)
    #22 0x7f23437e80b2 in __libc_start_main /build/glibc-ZN95T4/glibc-2.31/csu/../csu/libc-start.c:308:16
```

```
SUMMARY: AddressSanitizer: heap-use-after-free //boop/assorted_fuzzing/lib3mf/libzip/libzip-1.7.3/lib/zip_source_close.c:40:10 in
zip_source_close
Shadow bytes around the buggy address:
  0x0c0e7fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c0e7fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c0e7fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c0e7fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c0e7fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c0e7fff8000: fa fa fa fa fd fd fd fd fd fd[fd]fd fd fd fa fa
  0x0c0e7fff8010: fa fa fd fd fd fd fd fd fd fd fd fa fa fa fa fa
  0x0c0e7fff8020: fd fd fd fd fd fd fd fd fd fd fa fa fa fa 00 00
  0x0c0e7fff8030: 00 00 00 00 00 00 00 05 fa fa fa fa 00 00 00 00
  0x0c0e7fff8040: 00 00 00 00 00 fa fa fa fa fa 00 00 00 00 00 00
  0x0c0e7fff8050: 00 00 00 fa fa fa fa fa 00 00 00 00 00 00 00 00
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
==4692==ABORTING
```

Timeline

2021-01-14 - Vendor Disclosure
2021-03-10 - Public Release

CREDIT

Discovered by Lilith >_> of Cisco Talos.
