DCMTK - Bug #1209

Low severity short read in DcmMetaInfo::nextTagIsMeta()

2026-05-19 09:21 - Jörg Riesmeier

Status:	Closed	**Start date:**	2026-05-19
Priority:	High	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Library	**Estimated time:**	1:00 hour
Target version:	3.7.1		
Module:	dcmdata	**Compiler:**	
Operating System:			

Description

On 2026-05-17, Jiami (DMSAN) reported the following:

=== CUT ===

Dear DCMTK maintainers,

I found a use of uninitialized memory in DCMTK's file meta information parser while fuzzing the project with DMSAN, my sanitizer tool for detecting uses of uninitialized memory. The issue is in the dcmdata library code; I use dcmdump below only because it is a small official program that reaches the affected parser.

The root cause is in dcmdata/libsrc/dcmetinf.cc: DcmMetaInfo::nextTagIsMeta() peeks two bytes to decide whether the next element belongs to group 0002, but it does not check whether the stream actually returned two bytes:

```
char testbytes[2];
inStream.mark();
inStream.read(testbytes, 2);
inStream.putback();
return (testbytes[0] == 0x02 && testbytes[1] == 0x00) ||
       (testbytes[0] == 0x00 && testbytes[1] == 0x02);
```

DcmInputStream::read() returns the actual byte count. If the input has only one byte left, only testbytes[0] is initialized. testbytes[1] remains stale stack state, but it is still used by the group `0002` check. In the included PoC, an earlier real meta element leaves 0x00 in the same stack slot, and a final dangling byte 0x02 is then accepted as 02 00.

Native debugger observation of the two relevant calls shows:

```
ret=2 b0=02 b1=00 expr=1
ret=1 b0=02 b1=00 expr=1
```

The second line is the bug: one real input byte plus a stale stack byte is accepted as a group 0002 tag.

The attack scenario is a malformed DICOM file supplied to an import, validation, or inspection workflow that uses DCMTK. The victim application parses the file through dcmdata. The concrete effect is parser state control: a trailing single byte 0x02, which is not a complete DICOM tag, can be treated as a group 0002 file meta information tag when the missing byte comes from stale stack state. In my native reproduction, that changes the outcome from a successful parse of the control file to a parse failure for the PoC with I/O suspension or premature end of stream.

The same source pattern is visible as far back as the earliest public GitHub tag I checked, CAR96-3.0.1 from 1996-05-31, and remains present through DCMTK-3.7.0, the latest tag, and GitHub master (7354a5b, 2026-05-12). I directly reproduced the native behavior on Debian dcmtk 3.6.7 and on a build from current source.

I am not claiming code execution, OOB access, or information disclosure. From DMSAN's perspective, this is a use of uninitialized memory, which corresponds to CWE 457. I am flagging it as a security issue with low severity, while deferring to your judgment on the final classification.

My suggested fix is to check that the peek read returned exactly two bytes before comparing both bytes:

```
const offile_off_t bytesRead = inStream.read(testbytes, 2);
inStream.putback();
```

```
if (bytesRead != 2)
    return OFFalse;
```

I do not think initializing testbytes to zero is the right fix, because a trailing input byte 0x02 would then become 02 00 and would still be treated as group 0002.

The attached package contains a short report, two DICOM PoC files, a Dockerfile and helper script using native dcmdump, and the full patch. The reproduction does not depend on AFL, DMSAN, Valgrind, OSS Fuzz, or local paths.

[...]

History

#1 - 2026-05-19 09:28 - Jörg Riesmeier

- Description updated

#2 - 2026-05-19 09:32 - Jörg Riesmeier

- Description updated

#3 - 2026-05-26 09:38 - Jörg Riesmeier

- % Done changed from 0 to 100

- Estimated time set to 1:00 h

- Private changed from Yes to No

Fixed with commit e267be7ff.

#4 - 2026-05-26 09:38 - Jörg Riesmeier

- Status changed from New to Closed