DCMTK - Bug #1230

Potential DoS via mutlipe User Identity Negotiation sub-items

2026-06-19 17:11 - Michael Onken

Status:	Closed	**Start date:**	2026-06-19
Priority:	High	**Due date:**	
Assignee:	Michael Onken	**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:		**Compiler:**	
Operating System:			

Description

As reported by Arash Ale Ebrahim:

The DUL parser allows an A-ASSOCIATE-RQ PDU to contain multiple User Identity Negotiation sub-items (type 0x58). DICOM PS3.7 Annex D.3.3.7 defines this sub-item as occurring at most once per association, but the parser does not enforce that limit. Each parsed sub-item is heap-allocated with:

```
usrIdent =3D new UserIdentityNegotiationSubItemRQ();   // dulparse.cc:6=
08
    ...
    userInfo->usrIdent =3D usrIdent;                          // line 618
```

The previous pointer is silently overwritten with no free. Cleanup in destroyUserInformationLists() (helpers.cc:71) only deletes whichever pointer happens to be stored last, so every earlier allocation leaks permanently regardless of whether the association is accepted, rejected, or aborted. An unauthenticated peer can pack thousands of sub-items into a single PDU (bounded only by dcmAssociatePDUSizeLimit, default 1 MiB) and reconnect as often as it likes. Each 10-byte wire sub-item leaks ~56 bytes of heap (vtable + two OFStrings + payload pointers), giving roughly a 5-10x amplification ratio.

Reproduction (LeakSanitizer, storescp --fork --max-associations 1):

```
Direct leak of 1064 byte(s) in 19 object(s) allocated from:
        #0 operator new(unsigned long)
        #1 parseUserInfo          dcmnet/libsrc/dulparse.cc:608
        #2 parseAssociate         dcmnet/libsrc/dulparse.cc:265
        #3 AE_6_ExamineAssociateRequest  dcmnet/libsrc/dulfsm.cc:1216
        #4 PRV_StateMachine       dcmnet/libsrc/dulfsm.cc:810
        #5 DUL_ReceiveAssociationRQ      dcmnet/libsrc/dul.cc:716
        #6 ASC_receiveAssociation dcmnet/libsrc/assoc.cc:1797
        #7 acceptAssociation      dcmnet/apps/storescp.cc:1125
        #8 main                   dcmnet/apps/storescp.cc:1064
    SUMMARY: 1064 byte(s) leaked in 19 allocation(s).

20 sub-items 19 leaked (the last one is the only reachable pointer).

In an RSS-growth test (5000 sub-items 200 rounds):
        Baseline   :  15,868 KiB
        After PoC  : 108,964 KiB   (+93 MiB)
        60 s later : 108,964 KiB   (not reclaimed)
```

This affects every DCMTK server that calls ASC_receiveAssociation: storescp, dcmqrscp, dcmrecv, wlmscpfs, dcmpps, and any third-party SCP built on the DCMTK SCP API.

RCE assessment: NOT RCE

The pipeline tested whether the leaked heap blocks could be turned into a code execution primitive. They cannot. The allocations are UserIdentityNegotiationSubItemRQ objects placed on the general heap with no attacker control over their content after construction =E2=80=94 parseFromBuffer() validates and bounds-checks each f= ield, and the objects are never read again after the pointer is overwritten. There is no use-after-free, no type confusion, and no exploitable heap layout that emerges from the accumulation of identical, unreferenced objects. The impact is strictly memory exhaustion: the server runs out of addressable memory and is OOM-killed.

Suggested fix

Before overwriting userInfo->usrIdent, either reject the duplicate:

```
if (userInfo->usrIdent !=3D NULL) {
        DCMNET_WARN("Duplicate User Identity sub-item =E2=80=94 rejecting P=
DU");
        return makeDcmnetCondition(DULC_ILLEGALPDU, OF_error,
            "Duplicate User Identity Negotiation sub-item");
    }
```

or, for maximum compatibility with non-conformant peers, free first:

```
delete userInfo->usrIdent;
    userInfo->usrIdent =3D usrIdent;
```

History

#1 - 2026-06-19 17:24 - Michael Onken

Fixed with commit e61270.

#2 - 2026-06-26 07:56 - Michael Onken

- Status changed from New to Closed

- Private changed from Yes to No