DCMTK - Bug #1234

Remote unauthenticated memory-leak DoS in parseAssociate() presentation-context error path

2026-06-22 15:18 - Michael Onken

Status:	Closed	**Start date:**	2026-06-22
Priority:	High	**Due date:**	
Assignee:	Michael Onken	**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Target version:			
Module:	dcmnet	**Compiler:**	
Operating System:			

Description

As reported by Yuxiao Yan:

Summary

When parseAssociate() (dcmnet/libsrc/dulparse.cc) fails while parsing a Presentation Context item of an incoming A-ASSOCIATE-RQ/AC PDU, it releases the partially-parsed context with a plain free(context). That raw free() does NOT release context->transferSyntaxList (created by LST_Create()) nor the DUL_SUBITEM transfer-syntax entries already enqueued into it. The result is a heap memory leak triggerable by any remote peer during association negotiation — i.e. before any access control — leading to unbounded memory-exhaustion DoS.

This is the same class as CVE-2022-43272 (and CVE-2021-41687/41690). The extended-negotiation and user-information error branches of parseAssociate() were already hardened against this; the presentation-context branch was not.

Affected version

Current main, commit 5708ba6cd446d3e7b2cccc6c3a0bfb94b9aa5db9 (2026-06-11). The code path is long-standing; the surrounding error-path cleanup additions did not cover this branch.

Impact

- Attacker: remote, unauthenticated DICOM peer with TCP access to a DCMTK SCP that accepts associations (dcmqrscp, storescp, any ASC_receiveAssociation-based server). Reached during A-ASSOCIATE handling, prior to AE-title/access-control checks.
- Effect: each malformed A-ASSOCIATE PDU leaks the presentation context's transferSyntaxList head plus every transfer-syntax sub-item parsed before the triggering error. Amount per PDU scales with the number of transfer syntaxes packed before the failing sub-item; the default dcmAssociatePDUSizeLimit (1 MB) allows tens of thousands. No limit across reconnections, so repeated connections drive unbounded heap growth and eventual OOM / DoS.
- Suggested CVSS: AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H (~7.5, High), DoS only.

Root cause dcmnet/libsrc/dulparse.cc, parseAssociate(): case DUL_TYPEPRESENTATIONCONTEXTRQ:
case DUL_TYPEPRESENTATIONCONTEXTAC:
context = (PRV_PRESENTATIONCONTEXTITEM*)malloc(sizeof(PRV_PRESENTATIONCONTEXTITEM)); // ~234
if (context != NULL) {
(void) memset(context, 0, sizeof(*context));
cond = parsePresentationContext(type, context, buf, &itemLength, pduLength); // ~238
if (cond.bad()) {
free(context); // ~241 <-- leaks context->transferSyntaxList + sub-items
}
...

parsePresentationContext() does context->transferSyntaxList = LST_Create() (~406) and, for each transfer syntax sub-item, malloc + LST_Enqueue(&transferSyntaxList,…) (~450/458) BEFORE it may return a length error on a later/malformed sub-item. The free(context) frees only the struct.

Reproduction PoC scripts attached (raw sockets, no third-party deps):
1. ASAN harness linking the real dulparse.cc and calling parseAssociate()

(poc/fuzz_assoc.cc), or build dcmqrscp with -fsanitize=address.
2. Generate malformed PDU: python3 poc/poc_presctx_leak_amp.py /tmp/poc.bin 1500
— an A-ASSOCIATE-RQ whose presentation context holds 1 abstract syntax + N
valid transfer syntaxes followed by a sub-item whose length field exceeds the
remaining data, forcing parsePresentationContext() to return DULC_ILLEGALPDULENGTH.
3. Feed: ./fuzz_assoc /tmp/poc.bin — LeakSanitizer reports leaks at dulparse.cc:406
and :450; one 34 KB PDU leaks 168,032 bytes in 3001 allocations.
4. Against the real server: start dcmqrscp -c qr.cfg --single-process 11112, then
send the PDU on a raw socket repeatedly. Observed: +147 MB RSS over 400
malformed associations (monotonic), vs +22 MB (plateauing) for 400 well-formed. Suggested fix

Deep-free the presentation context on the error path instead of free(context): either add/extend a helper that dequeues+frees each DUL_SUBITEM in transferSyntaxList, then LST_Destroy(&transferSyntaxList), then free(context) (mirroring destroyAssociatePDUPresentationContextList() per node); or enqueue context into assoc->presentationContextList unconditionally and rely on the existing destroyAssociatePDUPresentationContextList(&assoc->presentationContextList) cleanup already run on cond.bad() (dulparse.cc ~294).

Thanks,

History

#1 - 2026-06-22 20:46 - Michael Onken

Closed with commit 688a16.

#2 - 2026-06-26 07:55 - Michael Onken

- Status changed from New to Closed

- Private changed from Yes to No