DCMTK - Bug #1232

Potential NUL-termination missing in use of strncpy

2026-06-19 17:20 - 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:	dcmnet	**Compiler:**	
Operating System:			

Description

As reported by Arash Ale Ebrahim:

Summary

The internal getString() helper used by every DIMSE command parser does:

```
strncpy(s, aString, maxlen);              // dimcmd.cc:209
    size_t s_len = strlen(s);                 // dimcmd.cc:221
    DU_stripLeadingAndTrailingSpaces(s);      // dimcmd.cc:224
```

The destination s is a DIC_UI (char[DIC_UI_LEN+1] = char[65]) and maxlen is DIC_UI_LEN (64). When getLength() maxlen, strncpy copies exactly 64 bytes without writing a terminating NUL, and s[64] is left in whatever state it was in before the call. The subsequent strlen() and DU_stripTrailingSpaces() then walk s as a C string; if s[64] is non-NUL, both read past the end of the field.

Today this is masked by the memset at dimcmd.cc:1885:

```
memset((char*)msg, 0, sizeof(*msg));
```

That memset pre-zeroes every byte of T_DIMSE_Message before any builder runs, so s[64] is always '\0' by the time strncpy finishes. The bug is latent but fragile: any future refactor that moves or removes the memset, replaces it with field-by-field initialization, or adds a new DIMSE service that bypasses DIMSE_parseCmdObject would immediately expose a real stack OOB read to any network peer that can send a DIMSE message.

ASAN reproduction (memset protection removed in PoC, simulating future refactor):

ERROR: AddressSanitizer: stack-buffer-overflow
READ of size 67 at 0x... thread T0
#0 __interceptor_strlen
#1 dimse_get_string   poc_getstring_no_nullterm.cc:63
Address at offset 674 in frame; next stack object at offset 720
('uid64') partially underflows.

Suggested fix (one line):

```
- strncpy(s, aString, maxlen);
    + OFStandard::strlcpy(s, aString, (size_t)maxlen + 1);
```

strlcpy always NUL-terminates. Every existing caller passes DIC_*_LEN against a DIC_*[*_LEN+1] buffer, so the +1 is always within bounds.

RCE assessment: NOT RCE (latent OOB read only)

The pipeline confirmed this is not exploitable today and has no RCE path even in the hypothetical future scenario where the memset is removed. The bug causes an overread in strlen() / DU_stripTrailingSpaces() — these functions walk a stack buffer and stop when they find a NUL byte; they perform no writes beyond their normal operation. A stack OOB read of this kind leaks adjacent stack bytes into internal DIMSE parser state (string length calculations), but those results are used only for whitespace stripping — they are not returned to the caller, written to a log, or placed on the wire. No code pointer, return address, or heap metadata is reachable from the overread region. Without a corresponding write primitive this cannot be developed into code execution.

History

#1 - 2026-06-19 17:26 - Michael Onken

Fixed in commit 782c652.

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

- Status changed from New to Closed

- Private changed from Yes to No