DCMTK - Bug #1231

Out-of-bounds read in OFStandard::sanitizeFilename()

2026-06-19 17:17 - 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 OFString overload of sanitizeFilename iterates by length() — not until NUL — and indexes the static charset table as sanitized_filename_charset[c - 32]. The guard condition is:

```
if (c != 0 && (c < 32 || c >= 127)) c = '_';
    else c = sanitized_filename_charset[c - 32];      // ofstd.cc:3426
```

When c == 0 (an embedded NUL byte in an OFString, which OFString supports by design), the c != 0 test fails, the else branch executes, and the index evaluates to sanitized_filename_charset[0 - 32] = sanitized_filename_charset[-32]. This is an out-of-bounds read 32 bytes before the start of the static array. The byte read from that address is written back into the filename string.

The char* overload at line 3432 uses while (*c) and stops at the first NUL — it is not vulnerable. The OFString overload is the one all DICOM-data callers use because DcmByteString preserves embedded NULs via length-based assignment.

The identical bug pattern exists in storescp.cc::sanitizeAETitle (line 2364) with its own sanitized_aetitle_charset table.

I want to flag something important about the commit history: this bug was introduced as a regression in commit edbb085e4 ("Sanitize all strings passed to the exec options", March 21 2026). That commit replaced the previous simple path-separator replacement with the charset-table approach. The C-string overload was written correctly (while (*c) terminates at NUL), but the OFString overload did not account for embedded NULs. All commits from edbb085e4 onward, including the current master, carry this bug.

ASAN reproduction (direct PoC, sanitized DCMTK build):

ERROR: AddressSanitizer: global-buffer-overflow
READ of size 1 at 0x... thread T0
#0 OFStandard::sanitizeFilename(OFString&) ofstd/libsrc/ofstd.cc:3426
#1 main poc_sanitizefilename_oob.cc:62
0x... is located 32 bytes to the left of global variable
'sanitized_filename_charset' defined in ofstd/libsrc/ofstd.cc:3408

On this build the adjacent .rodata byte is 0x3a (':'), so no path separator is injected here. On a different toolchain, optimization level, or DCMTK version the adjacent byte may differ — in particular a '/' (0x2f) would allow a network peer to inject a path separator into a storescp --sort-on-studyuid filename, potentially writing the incoming DICOM payload outside the configured destination directory (path traversal to arbitrary file write). That is the I/L component of the CVSS score.

Network-reachable path (confirmed with ASAN-instrumented storescp -su):

ERROR: AddressSanitizer: global-buffer-overflow
READ of size 1 at 0x... thread T0
#0 OFStandard::sanitizeFilename(OFString&) ofstd/libsrc/ofstd.cc:3426
#1 storeSCPCallback dcmnet/apps/storescp.cc:1986
#2 DIMSE_storeProvider dcmnet/libsrc/dimstore.cc:510
#3 storeSCP dcmnet/apps/storescp.cc:2264

A DICOM file whose StudyInstanceUID contains an embedded NUL (e.g. "1.2.3\x004") triggers the bug over the network via C-STORE. The embedded NUL survives DcmByteString's length-based extraction and reaches sanitizeFilename intact.

Suggested fix (one line, both functions):

```
- if (c != 0 && (c < 32 || c >= 127)) c = '_';
    + if (c < 32 || c >= 127) c = '_';
      else c = sanitized_filename_charset[c - 32];
```

Removing the c != 0 guard causes NUL bytes to be treated as control characters and replaced with '_', matching the intent of the original code.

RCE assessment: NOT RCE

The pipeline attempted to escalate this to code execution. It could not. The OOB read is from .rodata (the read-only data segment), which is mapped non-writable and non-executable. The attacker cannot choose the value of the byte that is read — they can only trigger the fixed -32 offset, so the output byte is determined entirely by whatever constant data the linker placed 32 bytes before sanitized_filename_charset. On our build that byte is 0x3a (':'), which is benign. Writing a constant byte from .rodata into an OFString provides no write-what-where primitive and no code-pointer corruption. The path traversal escalation path (if the byte were '/') would allow writing a DICOM file outside the destination directory, which is an integrity violation but not code execution in itself — a further prerequisite (e.g. a writable cron or service directory) would be needed and is outside the DCMTK threat model.

History

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

Fixed in commit e1f914 (and older commit e3878d).

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

- Status changed from New to Closed

- Private changed from Yes to No