



ESPILON
SECURITY RESEARCH

SECURITY ADVISORY

Out-of-bounds read in the BACnet/SC proprietary header-option decoder

CVE-2026-58265

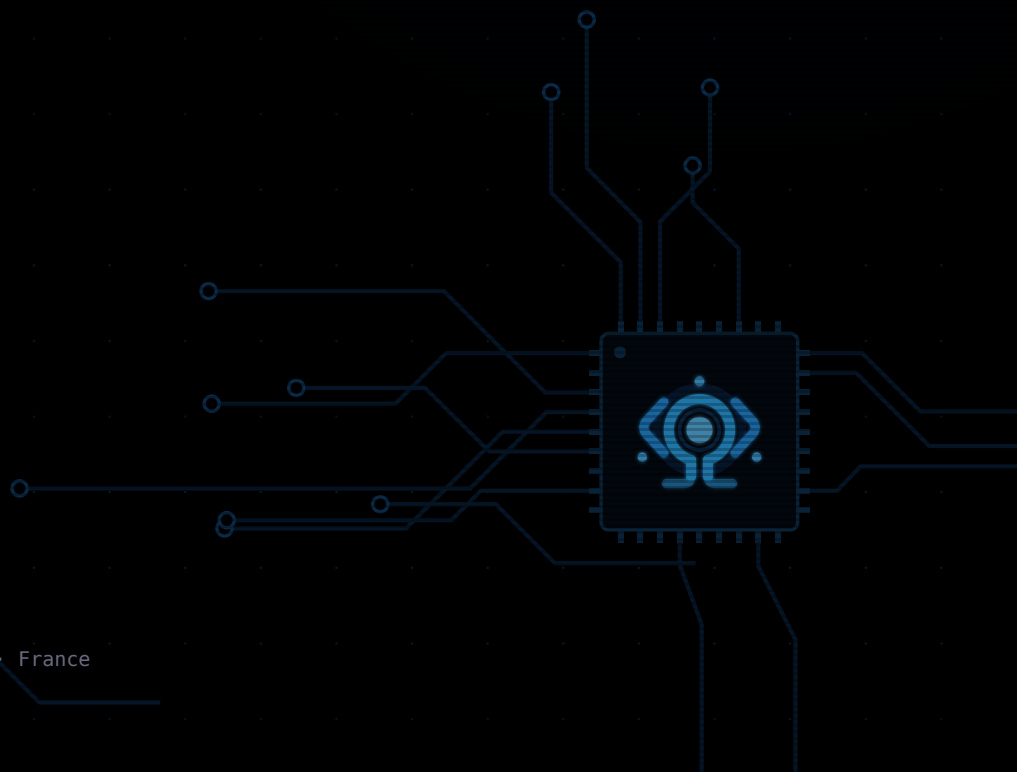
CVSS 6.5 · MEDIUM

bacnet-stack · BACnet/SC datalink

ADVISORY PUBLISHED 05 Jul 2026

Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France



01 Vulnerability

6.5

CVSS 3.1 BASE SCORE · GHSA-j66x-fr38-r7m5

AV:N / AC:L / PR:N / UI:N / S:U / C:L / I:N / A:L

MEDIUM

CVE	CVE-2026-58265
Product	bacnet-stack (open-source BACnet protocol stack in C), BACnet/SC datalink
Component	bvlc_sc_decode_proprietary_option() in src/bacnet/datalink/bsc/bvlc-sc.c
Affected	<= 1.5.x (master @ 4ed32fc), as stated by the advisory
Fixed	1.4.5 (backport 2ae6946) and 1.6.0 (PR #1396, merged 2026-06-19). The advisory also lists 1.5.1, but tag bacnet-stack-1.5.1 does not contain the fix.
Type	Out-of-bounds read (CWE-125)
Attacker	A BACnet/SC peer that can open a WebSocket connection to the node
Credit	Epsilon · Marouane El Moufid (Eun0us), reporter

Summary

BACnet/SC (Secure Connect) carries BACnet over TLS-protected WebSockets. Every BVLC-SC message can carry a list of header options, and bacnet-stack decodes a proprietary option in `bvlc_sc_decode_proprietary_option()`. The function reads the vendor identifier and option type at fixed offsets 3 to 5 of the option without checking the option's declared length, and the option-list validator accepts a proprietary option whose declared length is 0, which is 3 bytes long in total. A peer that ends a message with such an option makes the receiver read up to 3 bytes past the end of the message. On the Linux WebSocket server path that read can fall past the end of the heap allocation. The fix rejects proprietary options shorter than 3 bytes of header data and guards the reads.

The BACnet/SC datalink is compiled only when `BACDL_BSC` is enabled (`CMakeLists.txt:256, 352`); builds with only BACnet/IP or MS/TP do not contain this code.

02 Technical details

Vulnerable code

• `src/bacnet/datalink/bsc/bvlc-sc.c, bvlc_sc_decode_proprietary_option()` (bacnet-stack 4ed32fc)

```
584 static void bvlc_sc_decode_proprietary_option(  
585     uint8_t *in_options_list,  
586     uint16_t *out_vendor_id,  
587     uint8_t *out_proprietary_option_type,  
588     uint8_t **out_proprietary_data,  
589     size_t *out_proprietary_data_len)  
590 {  
591     uint16_t hdr_len;  
592  
593     *out_proprietary_data = NULL;  
594     *out_proprietary_data_len = 0;  
595     memcpy(&hdr_len, &in_options_list[1], sizeof(hdr_len));  
596     memcpy(out_vendor_id, &in_options_list[3], sizeof(uint16_t));  
597     *out_proprietary_option_type = in_options_list[5];  
598  
599     if (hdr_len > 3) {  
600         *out_proprietary_data = &in_options_list[6];  
601         *out_proprietary_data_len = hdr_len - 3;  
602     }  
603 }
```

The function's own comment (lines 571 to 572) says it must only be called on a previously validated option list. The validator, however, only checks that the declared length fits in the buffer:

• `src/bacnet/datalink/bsc/bvlc-sc.c, bvlc_sc_validate_options_headers()` (bacnet-stack 4ed32fc)

```
172     } else if (option == BVLC_SC_OPTION_TYPE_PROPRIETARY) {
173         if (!(flags & BVLC_SC_HEADER_DATA)) {
174             ....
179         }
180         options_len++;
181         if (options_len + 2 > option_headers_max_len) {
182             ....
188         }
189         memcpy(&hdr_len, &option_headers[options_len], 2);
190         options_len +=
191             2 /* header length */ + hdr_len /* length of data header */;
192     }
193
194     if (options_len > option_headers_max_len) {
```

Root cause

A proprietary option is a flags byte, a 2-byte header length, then header data that starts with a 2-byte vendor identifier and a 1-byte option type. The validator never requires the header length to be at least 3, so an option with `hdr_len = 0` is accepted as a complete 3-byte option. The decoder then reads offsets 3, 4 and 5 unconditionally (lines 596 and 597), before the only length test at line 599. When the option is the last thing in the message, those three bytes lie after the message.

Reachability

Header options are decoded inside `bvlc_sc_decode_message()`: `bvlc_sc_decode_dest_options_if_exists()` calls `bvlc_sc_decode_header_options()`, which calls the vulnerable function at line 1835. The `ADDRESS_RESOLUTION` branch rejects any payload (line 1981) and then decodes the destination options (line 1996), so the option is necessarily the end of the message.

On the server side, the first message on a new connection is decoded while the socket is in the awaiting-request state: `bsc_dispatch_srv_func()` calls `bsc_process_srv_awaiting_request()` (`bsc-socket.c:1180`), which calls `bvlc_sc_decode_message()` at line 836 and only afterwards checks whether the message is a Connect-Request (line 846). The defect is therefore reached before any BVLC-SC connection is accepted. The peer still has to complete the WebSocket and TLS handshake the node is configured to accept; the advisory rates the issue as requiring no privileges.

In the Linux port, `ports/linux/websocket-srv.c` reassembles each message in `fragment_buffer`, allocated at `BSC_RX_BUFFER_LEN` (512 bytes by default) and grown with `realloc()` to exactly the accumulated length when a message does not fit (lines 465 to 488), then passes that buffer to the dispatcher (line 503). A message larger than the current buffer is thus held in an allocation of its exact size, and the trailing over-read leaves the heap block. For a shorter message the read stays inside the buffer and returns stale bytes.

03 Impact and proof

Impact

The advisory scores low confidentiality and low availability impact. The up to 3 bytes read past the message are stored in the decoded option's `vendor_id` and `option_type` fields; whether they are ever sent back to a peer depends on the application, and no such path is claimed here. The advisory also names a crash of the node as a possible outcome, when the read reaches memory that is not mapped. There is no write primitive.

Proof

The dossier holds a host AddressSanitizer harness that includes the real `bvlc-sc.c` at commit 4ed32fc and calls the public `bvlc_sc_decode_message()` on a 7-byte `ADDRESS_RESOLUTION` message, allocated to its exact size, whose only destination option is a proprietary option with `hdr_len = 0`:

- poc_bacnet_sc.c, lines 19 to 32 (dossier, abridged)

```
size_t n = 7; /* exact message size, mimics realloc-to-exact recv buffer */
uint8_t *buf = (uint8_t *) malloc(n);
buf[0] = BVLC_SC_ADDRESS_RESOLUTION; /* no payload, decodes dest options */
buf[1] = BVLC_SC_CONTROL_DEST_OPTIONS; /* dest options present, no VADDR, no payload */
buf[2] = 0x12; buf[3] = 0x34; /* message id */
buf[4] = (uint8_t)(BVLC_SC_OPTION_TYPE_PROPRIETARY | BVLC_SC_HEADER_DATA); /* 0x1F|0x20, no MORE */
buf[5] = 0x00; buf[6] = 0x00; /* hdr_len = 0 -> option is exactly 3 bytes */
....
bool ok = bvlc_sc_decode_message(buf, n, &msg, &ec, &ed);
```

The archived sanitizer log (asan_output.txt) is not symbolized, as the symbolizer was unavailable in that run, and stops after five unnamed stack frames. Its opening lines report a 2-byte heap read out of bounds:

- asan_output.txt (dossier, lines 2 and 3, verbatim)

```
==239352==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x502000000017 at pc 0x55c17b561eaf bp 0x7fffee93d70 sp
0x7fffee93530
READ of size 2 at 0x502000000017 thread T0
```

The symbolized frames quoted in the advisory do not come from that log. We re-ran the same harness at 4ed32fc on 2026-10-28 (Debian clang 14.0.6, -fsanitize=address -g -O0) with the symbolizer available:

- AddressSanitizer, re-run of the dossier harness, 2026-10-28 (trimmed: pc/bp/sp, binary paths and build IDs, frames #6 to #8, allocation stack and shadow map removed)

```
==1368907==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000017
READ of size 2 at 0x602000000017 thread T0
#0 0x562e439ed476 in __asan_memcpy
#1 0x562e43a32f55 in bvlc_sc_decode_proprietary_option bvlc-sc.c:596:5
#2 0x562e43a32761 in bvlc_sc_decode_header_options bvlc-sc.c:1835:13
#3 0x562e43a2e9c3 in bvlc_sc_decode_dest_options_if_exists bvlc-sc.c:1856:9
#4 0x562e43a2c42d in bvlc_sc_decode_message bvlc-sc.c:1996:13
#5 0x562e43a317bf in main poc_bacnet_sc.c:32:15
0x602000000017 is located 0 bytes to the right of 7-byte region [0x602000000010,0x602000000017)
```

The 2-byte read is the memcpy of the vendor identifier at line 596, starting at offset 7 of the 7-byte message, the first byte after it.

04 Fix and recommendation

Upstream fix

The maintainer fixed both halves in PR #1396, merged on 2026-06-19 as commit 1068250. The validator now rejects a proprietary option with less than 3 bytes of header data, and the decoder zeroes its outputs and reads offsets 3 to 5 only when the header length allows it. A regression test injects an option with `hdr_len = 2` and expects `ERROR_CODE_HEADER_ENCODING_ERROR`.

```

• git show 1068250 -- src/bacnet/datalink/bsc/bvlc-sc.c

@@ -23,6 +23,8 @@ static const char *s_invalid_header_2 =
    "'Secure Path' header option must not have header data";
    static const char *s_invalid_header_3 =
    "'Proprietary Header' option must have header data";
+static const char *s_invalid_header_4 =
+    "'Proprietary Header' option header data length must be at least 3";
    static const char *s_result_incomplete =
    "BVL-Result message has incomplete payload";
    static const char *s_result_incorrect_bvlc_function =
@@ -187,6 +189,12 @@ static bool bvlc_sc_validate_options_headers(
    return false;
    }
    memcpy(&hdr_len, &option_headers[options_len], 2);
+    if (hdr_len < 3) {
+        *error_code = ERROR_CODE_HEADER_ENCODING_ERROR;
+        *error_class = ERROR_CLASS_COMMUNICATION;
+        *error_desc_string = s_invalid_header_4;
+        return false;
+    }
    options_len +=
        2 /* header length */ + hdr_len /* length of data header */;
@@ -592,9 +600,14 @@ static void bvlc_sc_decode_proprietary_option(
    *out_proprietary_data = NULL;
    *out_proprietary_data_len = 0;
+    *out_vendor_id = 0;
+    *out_proprietary_option_type = 0;
    memcpy(&hdr_len, &in_options_list[1], sizeof(hdr_len));
-    memcpy(out_vendor_id, &in_options_list[3], sizeof(uint16_t));
-    *out_proprietary_option_type = in_options_list[5];
+    if (hdr_len >= 3) {
+        memcpy(out_vendor_id, &in_options_list[3], sizeof(uint16_t));
+        *out_proprietary_option_type = in_options_list[5];
+    }

    if (hdr_len > 3) {
        *out_proprietary_data = &in_options_list[6];

```

Recommendation

Update to bacnet-stack 1.4.5, 1.6.0 or later, or apply commit 1068250 to a vendored copy of bvlc-sc.c. Users of the 1.5 line need the patch: the advisory lists 1.5.1 as fixed, but tag bacnet-stack-1.5.1 (commit 3a74c74) and the bacnet-stack-1.5 branch as of 2026-10-28 still carry the unguarded reads. Products that do not build the BACnet/SC datalink are not exposed. Restricting which peers can complete the TLS handshake with a BACnet/SC hub or node reduces who can reach the decoder, but does not fix it.

05 Disclosure

Timeline

DATE (UTC)	EVENT
2026-06-16	Defect verified on bacnet-stack master 4ed32fc; private security advisory GHSA-j66x-fr38-r7m5 opened by the reporter
2026-06-19	Fix merged upstream (PR #1396, commit 1068250)
2026-07-05	bacnet-stack 1.4.5 release commit, containing the backport 2ae6946; advisory GHSA-j66x-fr38-r7m5 published with CVE-2026-58265
2026-07-13	bacnet-stack 1.6.0 release commit, containing 1068250

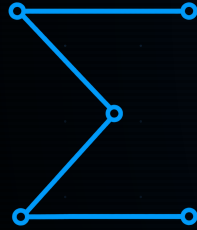
Credit

Found and reported by Marouane El Moufid (EunOus) of Epsilon, credited as reporter in the advisory. The fix was developed by Steve Karg (skarg), credited as remediation developer.

References

Advisory	github.com/bacnet-stack/bacnet-stack/security/advisories/GHSA-j66x-fr38-r7m5
CVE record	cve.org/CVERecord?id=CVE-2026-58265
Fix	github.com/bacnet-stack/bacnet-stack/pull/1396
Fix commit	github.com/bacnet-stack/bacnet-stack/commit/1068250f197c0cc78b6357374d8be5df7d94b3cc

On 2026-10-28 the CVE Services API still listed CVE-2026-58265 as RESERVED, with no published record; the GitHub advisory is the authoritative public source.



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 • France