ESPILON
SECURITY RESEARCH

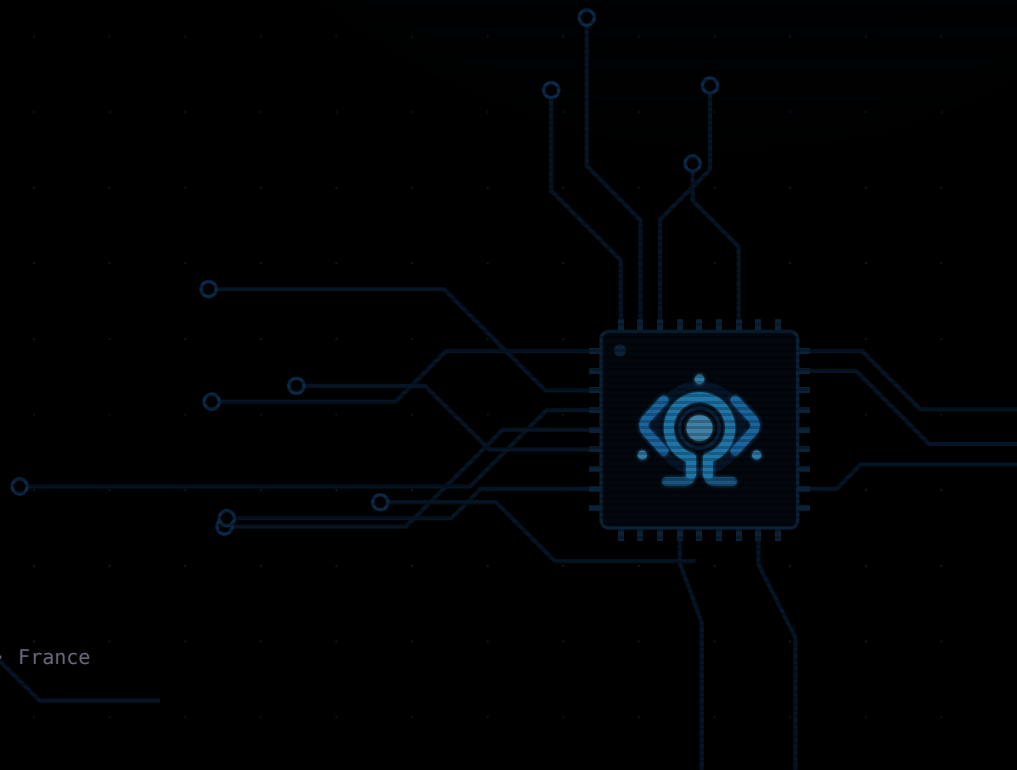
SECURITY ADVISORY

Two out-of-bounds writes in the SOES EtherCAT slave stack

CVE-2026-129794 · CVE-2026-129796 CVSS 8.8 · HIGH

RT-Labs SOES · EoE fragment reassembly and CoE SDO download

ADVISORY PUBLISHED **19 Nov 2026**



Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France

01 Vulnerabilities

8.8

HIGH

CVSS 3.1 BASE SCORE · RT-LABS (CNA)

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

Same score and vector for both CVEs, as published by RT-Labs in RRTL-261119-01 and in the two CVE records.

CVE	CVE-2026-129794 (EoE) and CVE-2026-129796 (CoE), RT-Labs advisory RRTL-261119-01, 2026-11-19
Product	SOES (Simple Open EtherCAT Slave), EtherCAT slave stack by RT-Labs, published as `OpenEtherCATsociety/SOES`
Component	`EOE_receive_fragment()` in `soes/esc_eoe.c` `SDO_download()` in `soes/esc_coe.c`
Type	129794: heap out-of-bounds write (CWE-787). 129796: integer underflow (CWE-191) leading to an out-of-bounds write (CWE-787)
Attacker	A malicious or compromised EtherCAT master, or any device that can send mailbox frames to the slave on the segment. No authentication at this layer
Fix	SOES v3.2.1 (commits `8ec115d` and `54a7404`)
Credit	Brian Willows (graith.co.uk), Eun0us (Espilon), f0rw4rd (quellsec.dev): shared credit, see section 05

Summary

SOES is an open-source EtherCAT slave stack. It runs on the microcontroller of an EtherCAT device and handles the mailbox protocols the master uses for acyclic exchanges, among them CoE (CANopen over EtherCAT) for the object dictionary and EoE (Ethernet over EtherCAT) to tunnel Ethernet frames. Both issues sit on this mailbox path, and both have the same shape: a length taken from the master's frame is used as a bound without being checked against the real size of the destination.

- **CVE-2026-129794.** EoE reassembly copies each fragment into the receive buffer the application supplied, but bounds the copy by a frame size read from the first fragment's header (up to 2016 bytes), not by the buffer's capacity. With the 592-byte lwIP pool buffer of the stack's own EoE example, the master writes bytes of its choice past the end of the buffer.
- **CVE-2026-129796.** SDO download to a flexible-length object pads a `VISIBLE_STRING` with `memset(data + size, 0, actsize - size)`, with no check that `size <= actsize`. A download size larger than the object makes the unsigned subtraction wrap, and `memset` runs for about 4 GiB from a point already past the object.

RRTL-261119-01 groups 21 CVEs in SOES (CVE-2026-129794 to CVE-2026-129814), reported by several researchers. Espilon is credited on two of them, and this report covers only those two.

Affected and fixed versions

Per RT-Labs and the CVE records: SOES before 3.2.1, that is, up to and including v3.2.0. We checked that the vulnerable code is the same in v3.2.0 as in commit `6ef7b94`, the one our reports were written against.

CVE	COMPONENT	AFFECTED	FIXED
CVE-2026-129794	`esc_eoe.c`, EoE receive	< 3.2.1	v3.2.1, 8ec115d
CVE-2026-129796	`esc_coe.c`, SDO download	< 3.2.1	v3.2.1, 54a7404

Timeline

DATE	EVENT
2026-06-17	Espilon reports the CoE SDO download underflow to RT-Labs, with a host ASan harness
2026-07-09	Espilon reports the EoE reassembly overflow to RT-Labs, with a host ASan harness

DATE	EVENT
2026-10-22	SDO download fix authored at RT-Labs (`54a7404`)
2026-10-23	EoE reassembly fix authored at RT-Labs (`8ec115d`)
2026-11-16	CVE IDs reserved by RT-Labs (CNA), draft advisory shared with the reporters
2026-11-19	Espilon approves the draft advisory, the credit and the publication date
2026-11-19	Advisory RRTL-261119-01 published, SOES v3.2.1 tagged
2026-11-20	SOES v3.2.1 release published on GitHub
2026-11-21	CVE records published

The report dates and the draft review are from our own disclosure records. The other dates are public: the advisory, the CVE records and the SOES git history.

02 EoE fragment reassembly (CVE-2026-129794)

Vulnerable code

soes/esc_eoe.c at 6ef7b94 (same code in v3.2.0), EOE_receive_fragment()

```
755     /* Start of new frame at fragment 0 */
756     if(EOEvar.rxfragmentno == 0)
757     {
758        EOEvar.rxframesize = (EOE_HDR_FRAME_OFFSET_GET(frameinfo2) << 5);
759
760        if(EOEvar.rxebuf.payload != NULL)
761        {
762           EOEvar.rxebuf.len = EOEvar.rxframesize;
763           EOEvar.rxframeoffset = 0;
764           EOEvar.rxframeno = EOE_HDR_FRAME_NO_GET(frameinfo2);
765        }
...
794     /* Check so allocated buffer is sufficient */
795     if ((EOEvar.rxframeoffset + eoedatasize) <= EOEvar.rxframesize)
796     {
797        memcpy((uint8_t *)(EOEvar.rxebuf.payload + EOEvar.rxframeoffset),
798              eoembx->data,
799              eoedatasize);
800        EOEvar.rxframeoffset += eoedatasize;
801        EOEvar.rxfragmentno++;
802     }
```

Root cause

An EoE frame arrives in fragments, one per mailbox frame. On fragment 0, the 6-bit frame-offset field of the header is read as the total frame size: `rxframesize` is that field shifted left by 5, so at most `63 << 5`, 2016 bytes (line 758). Line 762 then overwrites `rxebuf.len`, which the application had set to the real capacity of its buffer, with this value from the wire. The check on line 795 compares the running offset plus the fragment length against `rxframesize`, a number the master chose, and the `memcpy` on line 797 copies the fragment data into the buffer.

The buffer comes from the application, through the `get_buffer()` callback, at a spot the code already marks as unchecked:

soes/esc_eoe.c at 6ef7b94, EOE_init_rx()

```
935     /* Fetch buffer */
936     if(EOEvar.rxebuf.payload == NULL)
937     {
938        if(eoe_cfg->get_buffer != NULL)
939        {
940           /* TODO: verify size VS buffer size */
941           eoe_cfg->get_buffer(&EOEvar.rxebuf);
942        }
943     }
```

In the EoE example shipped with the stack, `applications/rtl_lwip_eoe`, the callback returns a fixed lwIP pool buffer, `pbuf_alloc(PBUF_RAW, PBUF_POOL_BUFSIZE, PBUF_POOL)` (`main.c:113`), whose size does not depend on the frame: 592 bytes with the default lwIP settings. Any sequence of fragments that takes the offset past 592 while staying under the declared 2016 bytes passes the check and is copied past the end of the buffer.

Reachability

EoE is compiled in by default (`USE_EOE 1` in `soes/options.h`), and `rtl_lwip_eoe` enables it. An EoE mailbox frame goes from `ESC_eoeprocess()`, the public receive entry, to `EOE_receive_fragment()` (`esc_eoe.c:1027`) as soon as the slave's mailbox is running. No object-dictionary entry and no prior configuration are needed. The condition is that the application hands over a receive buffer smaller than the 2016 bytes the wire can declare, which is the case of the shipped example. Applications built with `USE_EOE 0` do not have this path.

Impact

The master chooses both the content and the length of the write: a linear overflow of up to 1424 bytes past the 592-byte buffer (2016 minus 592), delivered fragment by fragment, into the memory that follows the buffer, which in the example is lwIP pool memory. The expected result is corrupted memory on the slave, then a crash or a hang. RT-Labs scores it `C:H/I:H/A:H` and lists "denial of service or potential code execution". We did not demonstrate code execution on a physical slave, and this report makes no code-execution claim: what we demonstrated is the write itself, with its content and its length under the master's control.

Proof

A host harness compiles the unmodified `esc_eoe.c` of `6ef7b94` and drives the public entry `ESC_eoeprocess()`: the length check, the size arithmetic and the copy run inside the real code. Its `get_buffer()` follows `rtl_lwip_eoe`: a fixed 592-byte buffer that reports its real length. Only `ESC_claimbuffer()`, on the transmit side, is stubbed. One mailbox frame carries fragment 0 with the frame-offset field at 63 (2016 bytes) and 1024 bytes of data. Output of our report of 2026-07-09, trimmed:

• AddressSanitizer: WRITE in EOE_receive_fragment (one 1024-byte fragment), trimmed

```
[harness] dest pbuf capacity=592, wire rxframesize=2016
==692318==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6160000002d0
WRITE of size 1024 at 0x6160000002d0 thread T0
    #0 in __asan_memcpy
    #1 in EOE_receive_fragment soes/esc_eoe.c:797:7
    #2 in ESC_eoeprocess soes/esc_eoe.c:1027:16
0x6160000002d0 is located 0 bytes to the right of 592-byte region
allocated by thread T0 here:
    #1 in appl_get_buffer harness_eoe.c:60:24
    #2 in EOE_init_rx soes/esc_eoe.c:941:10
```

The 1024-byte fragment keeps the proof short. The example application uses 128-byte mailboxes (`MBXSIZE 128`), so on 2026-11-23 we re-ran the harness with that size: seven 96-byte fragments of the same frame, the seventh starting at offset 576, give the same report (`WRITE of size 96`, 0 bytes after the 592-byte region). A build without sanitizer, with the buffer placed against a guard page, stops with `SIGSEGV` on the first byte past the buffer.

03 CoE SDO download (CVE-2026-129796)

Vulnerable code

• soes/esc_coe.c at 6ef7b94 (same lines in v3.2.0), SDO_download(), abridged

```
812                /* normal download */
813                size = (etohl (coesdo->size) & 0xffff);
814                mbxdata = (&(coesdo->size)) + 1;
815             }
816             actsize = BITS2BYTES((objd + nsub)->bitlength);
817             if (actsize != size)
818             {
819                /* entries with data types VISIBLE_STRING, OCTET_STRING,
820                 * UNICODE_STRING, ARRAY_OF_INT, ARRAY_OF_SINT,
821                 * ARRAY_OF_DINT, and ARRAY_OF_UDINT may have flexible length
822                 */
823                uint16_t type = (objd + nsub)->datatype;
824                if (type == DTYPE_VISIBLE_STRING)
825                {
826                   /* pad with zeroes up to the maximum size of the entry */
827                   memset((objd + nsub)->data + size, 0, actsize - size);
828                }
...
864                copy2mbx (mbxdata, (objd + nsub)->data, size);
```

Root cause

In a normal (non-expedited) download, `size` is the size field of the master's request, kept to 16 bits (up to 65535, line 813). `actsize` is the capacity of the object in the dictionary (line 816). When the two differ and the object is a `VISIBLE_STRING`, line 827 pads the end of the string with zeros, with a length of `actsize - size`. Both are `uint32_t` and nothing checks `size <= actsize` first. For a 16-byte object and a size of 100, the length wraps to `0xFFFFFFAC`, 4294967212 bytes, and the destination `data + size` is already 84 bytes past the object.

The other flexible-length types (`OCTET_STRING`, `UNICODE_STRING`, the `ARRAY_OF_*` types) skip the padding and reach `copy2mbx()` on line 864, which copies `size` bytes, capped only at what the mailbox carries, into the object, again with no check against `actsize`. This is the second write covered by the CVE: RT-Labs describes up to about 116 bytes of master data written into an object smaller than a mailbox payload. Our report demonstrated the `memset` and recommended the same guard for this copy; the copy itself is described here from the RT-Labs advisory.

Reachability

The master needs a writable flexible-length entry in the slave's object dictionary. The SOES demo applications declare their string objects read-only, so they are not affected as shipped. Writable strings and arrays (a configurable device name, parameter strings) are a common setup in real devices, and SOES is a library that each product configures. No credential is involved: any master can send an SDO download.

Impact

The `memset` writes zeros, not attacker data, but its length is about 4 GiB, so it runs over whatever follows the object until it reaches unmapped memory: the slave crashes, after its memory past the object has been overwritten. The `copy2mbx()` path writes bytes chosen by the master. RT-Labs scores the CVE `C:H/I:H/A:H` and lists "memory corruption or denial of service; potential code execution". We demonstrated the `memset` write and make no code-execution claim.

Proof

The harness of our report of 2026-06-17 compiles the real `esc_coe.c` of `6ef7b94` (included in the harness, since `SDO_download()` is static), declares one writable 16-byte `VISIBLE_STRING` object, and sends a normal download with a size of 100. The report's output and our re-run of 2026-11-23 agree; the re-run, trimmed:

- AddressSanitizer: WRITE in SDO_download (16-byte object, size 100), trimmed

```
==993129==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000074
WRITE of size 4294967212 at 0x602000000074 thread T0
    #0 in __asan_memset
    #1 in SDO_download soes/esc_coe.c:827:19
    #2 in main poc_soes_real.c:62:4
0x602000000074 is located 84 bytes to the right of 16-byte region
```

A second harness, written on 2026-10-26, enters through the public CoE dispatcher `ESC_coeprocess()` with the command byte a master sends (`0x21`, download request with size indicator), and keeps the object inside a `.bss` structure, as SOES applications do. Run with a size of 100 on 2026-11-23, it ends in a write to unmapped memory from `SDO_download` (`esc_coe.c:827`), called by `ESC_coeprocess` (`esc_coe.c:1501`).

04 Fix and recommendation

Upstream fixes

Both fixes are by RT-Labs and ship in SOES v3.2.1. For EoE, commit `8ec115d` no longer overwrites `rxebuf.len` with the wire value and clamps `rxframesize` to it, so the check before the copy now compares against the real capacity of the buffer:

OpenEtherCATsociety/SOES commit 8ec115d1ba8b, soes/esc_eoe.c

```
@@ -759,7 +759,11 @@ static void EOE_receive_fragment (void)

       if(EOEvar.rxebuf.payload != NULL)
       {
-         EOEvar.rxebuf.len = EOEvar.rxframesize;
+         /* Limit frame size to the real buffer capacity */
+         if (EOEvar.rxframesize > EOEvar.rxebuf.len)
+         {
+            EOEvar.rxframesize = EOEvar.rxebuf.len;
+         }
          EOEvar.rxframeoffset = 0;
          EOEvar.rxframeno = EOE_HDR_FRAME_NO_GET(frameinfo2);
       }
```

For CoE, commit `54a7404` aborts the transfer with `ABORT_TYPEMISMATCH` when the declared size is larger than the object, before the padding and before `copy2mbx()`, and only pads when the size is smaller:

OpenEtherCATsociety/SOES commit 54a740490b02, soes/esc_coe.c

```
@@ -814,7 +814,12 @@ static void SDO_download (void)
                mbxdata = (&(coesdo->size)) + 1;
             }
             actsize = BITS2BYTES((objd + nsub)->bitlength);
-            if (actsize != size)
+            if (size > actsize)
+            {
+               set_state_idle (0, index, subindex, ABORT_TYPEMISMATCH);
+               return;
+            }
+            if (size < actsize)
             {
```

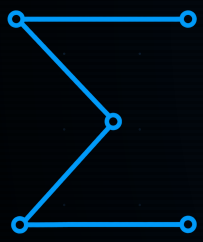
On 2026-11-23 we ran our four harnesses (EoE with one 1024-byte fragment, EoE with 96-byte fragments, SDO download called directly and through `ESC_coeprocess()`) against v3.2.0 and v3.2.1. All four fault on v3.2.0 and run to completion without any sanitizer report on v3.2.1.

Recommendation

Update SOES to v3.2.1 or later. SOES is compiled into the device firmware, so the fix reaches devices only through a firmware update from the device vendor. RT-Labs also advises keeping EtherCAT networks isolated from untrusted devices and away from public networks. Until a device is updated: firmware that does not use EoE can be built with `USE_EOE 0` an EoE application whose `get_buffer()` always returns at least 2016 bytes cannot be overrun, since the declared frame size never exceeds that; and keeping flexible-length dictionary entries read-only removes the SDO download path.

05 Credit and references

- Credit is shared. RT-Labs credits CVE-2026-129794 and CVE-2026-129796 to Brian Willows (graith.co.uk), Eun0us (Espilon) and f0rw4rd (quellsec.dev), in the advisory and in both CVE records. This report describes our part only: our two reports, our harnesses and our dates.
- Espilon: Eun0us (Marouane El Moufid). Thanks to RT-Labs for coordinating the disclosure with all the reporters.
- RT-Labs advisory RRTL-261119-01, 2026-11-19: docs.rt-labs.com/security/advisories/rrtl-261119-01.html
- Fixes: SOES commit 8ec115d (EoE), SOES commit 54a7404 (CoE)
- Release: SOES v3.2.1, github.com/OpenEtherCATsociety/SOES/releases/tag/v3.2.1
- CVE records: www.cve.org/CVERecord?id=CVE-2026-129794, www.cve.org/CVERecord?id=CVE-2026-129796



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 · France