



ESPILON
SECURITY RESEARCH

SECURITY ADVISORY

Out-of-bounds read in the Router Advertisement option parser

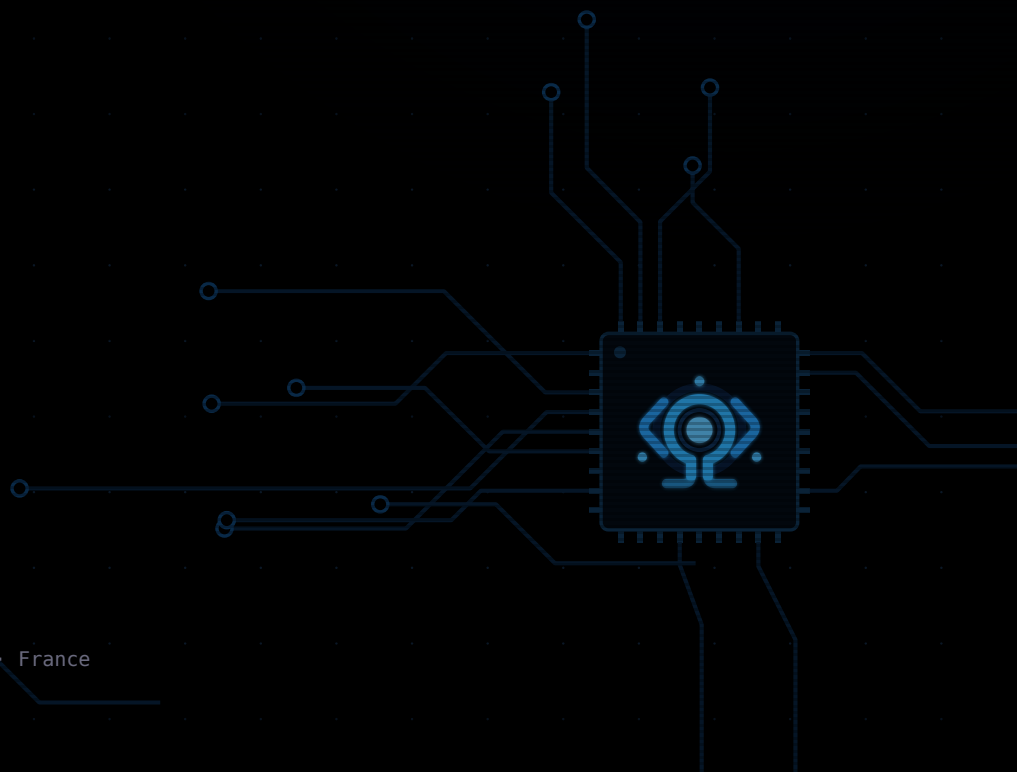
CVE-2026-7425 **CVSS 6.5 • MEDIUM**

AWS • FreeRTOS-Plus-TCP

CVE PUBLISHED **29 Apr 2026**

Marouane El Moufid (Eun0us)

espilon.net • Association loi 1901 • France



01 Vulnerability

6.5

CVSS 3.1 BASE SCORE · AMAZON (CNA)

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

CVSS 4.0 (CVE record): 6.0 MEDIUM · AV:A/AC:L/AT:P/PR:N/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:N

MEDIUM

CVE	CVE-2026-7425
Advisory	GHSA-gffr-xgig-jh9j
Component	FreeRTOS-Plus-TCP, vReceiveRA_ReadReply() and vReceiveRA() in source/FreeRTOS_RA.c
Affected	V4.0.0 to V4.2.5, V4.3.0 to V4.4.0
Fixed	V4.2.6 and V4.4.1
Type	Out-of-bounds read (CWE-125)
Configuration	ipconfigUSE_IPv6 and ipconfigUSE_RA (both enabled by default)
Credit	Epsilon · Marouane El Moufid (Eun0us)

Summary

When FreeRTOS-Plus-TCP parses the options of a Router Advertisement (RA), it maps a Prefix Information option onto the 32-byte `ICMPPrefixOption_IPv6_t` structure after checking only that the option's own declared length fits in the frame. That declared length can be as small as 8 bytes. A short Prefix Information option placed at the end of an RA makes the stack read the prefix lifetimes and the 16-byte prefix from up to 24 bytes past the end of the received data. An unauthenticated attacker on the same link triggers it with a single crafted RA.

This advisory is distinct from CVE-2026-7426 (GHSA-97qg-4359-xm3x), an out-of-bounds *write* driven by the Prefix *Length* value in the same option (fix ee62f80). The defect here is the option's *size* and the *read* it allows (fix d210ba0).

02 Technical details

Root cause

`vReceiveRA_ReadReply()` walks the options that follow the fixed RA header. Each option carries a length in units of 8 bytes; the loop rejects zero and rejects an option that would run past `uxLast`, the end of the received data:

```
• source/FreeRTOS_RA.c, vReceiveRA_ReadReply() (V4.4.0)

270 static ICMPPrefixOption_IPv6_t * vReceiveRA_ReadReply( const NetworkBufferDescriptor_t * pxNetworkBuffer )
271 {
272     size_t uxIndex = 0U;
273     const size_t uxICMPSize = sizeof( ICMPRouterAdvertisement_IPv6_t );
274     const size_t uxNeededSize = ipSIZE_OF_ETH_HEADER + ipSIZE_OF_IPv6_HEADER + uxICMPSize;
275     /* uxLast points to the first byte after the buffer. */
276     const size_t uxLast = pxNetworkBuffer->xDataLength - uxNeededSize;
277     uint8_t * pucBytes = &( pxNetworkBuffer->pucEthernetBuffer[ uxNeededSize ] );
278     ICMPPrefixOption_IPv6_t * pxPrefixOption = NULL;
279
280     while( ( uxIndex + 1U ) < uxLast )
281     {
282         uint8_t ucType = pucBytes[ uxIndex ];
283         size_t uxPrefixLength = ( size_t ) pucBytes[ uxIndex + 1U ];
284         size_t uxLength = uxPrefixLength * 8U;
285
286         if( uxPrefixLength == 0U )
287         {
288             /* According to RFC 4861, length of the option value 0 is invalid. Hence returning from here */
289             FreeRTOS_printf( ( "RA: Invalid length of the option value as zero. " ) );
290             break;
291         }
292
293         if( uxLast < ( uxIndex + uxLength ) )
294         {
295             FreeRTOS_printf( ( "RA: Not enough bytes ( %u > %u )\n", ( unsigned ) ( uxIndex + uxLength ),
296                               ( unsigned ) uxLast ) );
297             break;
298         }
299     }
```

These checks are relative to the length the option *declares*. For a Prefix Information option, the code then casts the option bytes to the full structure without comparing that length to `sizeof( ICMPPrefixOption_IPv6_t )`:

```
• source/FreeRTOS_RA.c, vReceiveRA_ReadReply(), Prefix Information case (V4.4.0)

314 case ndICMP_PREFIX_INFORMATION: /* 3 */
315     /* MISRA Ref 11.3.1 [Misaligned access] */
316     /* More details at: https://github.com/FreeRTOS/FreeRTOS-Plus-TCP/blob/main/MISRA.md#rule-113 */
317     /* coverity[misra_c_2012_rule_11_3_violation] */
318     pxPrefixOption = ( ( ICMPPrefixOption_IPv6_t * ) &( pucBytes[ uxIndex ] ) );
319
320     FreeRTOS_printf( ( "RA: Prefix len %d Life %u, %u (%pip)\n",
321                     pxPrefixOption->ucPrefixLength,
322                     ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulValidLifeTime ),
323                     ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulPreferredLifeTime ),
324                     ( void * ) pxPrefixOption->ucPrefix ) );
325     break;
```

The structure (`FreeRTOS_IPv6_Private.h:201`) is 32 bytes: type, length, prefix length and flags in the first 4 bytes, then `ulValidLifeTime` at offset 4, `ulPreferredLifeTime` at 8, `ulReserved` at 12 and `ucPrefix[ 16 ]` at 16 to 31. With a declared length of 1 (8 bytes) and the option ending exactly at `uxLast`, the check at line 293 passes, yet every field from offset 8 onward lies past the received data: `ulPreferredLifeTime`, `ulReserved` and the 16-byte `ucPrefix`, up to 24 bytes in total.

Where the bytes are read

The log call at lines 320 to 324 reads them when `FreeRTOS_printf` is enabled. In every build, the pointer is returned to `vReceiveRA()`, which copies the fields into each end-point that is waiting for an RA:

```
• source/FreeRTOS_RA.c, vReceiveRA() (V4.4.0)

406 if( ( pxEndPoint->bits.bWantRA != ipFALSE_BOOL ) && ( pxEndPoint->xRADATA.eRAState == eRAStateWait ) )
407 {
408     pxEndPoint->ipv6_settings.uxPrefixLength = pxPrefixOption->ucPrefixLength;
409     ( void ) memcpy( pxEndPoint->ipv6_settings.xPrefix.ucBytes, pxPrefixOption->ucPrefix,
410                     ipSIZE_OF_IPv6_ADDRESS );
411     ( void ) memcpy( pxEndPoint->ipv6_settings.xGatewayAddress.ucBytes, pxICMPPacket-
412     >xIPHeader.xSourceAddress.ucBytes, ipSIZE_OF_IPv6_ADDRESS );
413     pxEndPoint->xRADATA.bits.bRouterReplied = ipTRUE_BOOL;
414     pxEndPoint->xRADATA.uxRetryCount = 0U;
415     pxEndPoint->xRADATA.ulPreferredLifeTime = FreeRTOS_ntohl( pxPrefixOption->ulPreferredLifeTime );
```

Line 409 copies the 16 bytes of `ucPrefix` (option offsets 16 to 31) and line 414 reads `ulPreferredLifeTime` (offsets 8 to 11). The out-of-range bytes become the end-point's network prefix and preferred lifetime, and the state machine goes on to build an address from that prefix.

Whether the read crosses the end of the heap block depends on the network buffer scheme. With `BufferAllocation_2.c`, `pxGetNetworkBufferWithDescriptor()` allocates with `pvPortMalloc()` the requested size plus 2 bytes, rounded up to a multiple of `sizeof( size_t )`, plus `ipBUFFER_PADDING` in front. When a driver requests a receive buffer of the frame's length, at most a few bytes of slack follow the frame, so most of the 24 bytes lie past the end of the heap block (derived from the code, not observed). With `BufferAllocation_1.c` (fixed-size buffers) the bytes come from the unused tail of the buffer.

Attacker position and preconditions

- Adjacent network: one ICMPv6 RA (type 134) on the device's link, with a valid ICMPv6 checksum and a non-zero router lifetime (`FreeRTOS_RA.c:392`). No authentication, no prior exchange.
- `ipconfigUSE_IPv6` and `ipconfigUSE_RA` compiled in, both enabled by default in `FreeRTOSIPConfigDefaults.h`.
- The parse in `vReceiveRA_ReadReply()` runs for any accepted RA; the copy in `vReceiveRA()` additionally needs an end-point configured for SLAAC (`bWantRA`) and waiting in `eRAStateWait`.

03 Impact and evidence

Impact. The established effect is an out-of-bounds read of up to 24 bytes past the received RA, whose content then replaces the end-point's prefix and preferred lifetime. The CNA rates availability High and confidentiality and integrity None. We make no claim beyond that.

Evidence. The libFuzzer harness of our March 2026 campaign on `vReceiveRA()` never reached the copy: its routing stub returned no end-point, so the end-point loop at line 402 had nothing to update. On 2026-10-28 we rebuilt that harness against the same vulnerable tree (af6b379, 2026-03-16) with one change, an IPv6 end-point with `bWantRA` set, waiting

in `eRAStateWait` and bound to the interface. The functions called after the copy are stubbed to abort, so they cannot hide a result. The frame was built from the code: 1,500 bytes, filling the harness's 1,500-byte receive buffer, a non-zero router lifetime, one unknown option of 1,416 bytes, then a Prefix Information option with a length field of 1 ending 6 bytes before the frame end. Those 6 trailing bytes are an option of length 0, which ends the loop.

- AddressSanitizer, `FreeRTOS_RA.c` at `af6b379`, 1,500-byte RA with a short final Prefix Information option (2026-10-28)

```
==2761117==ERROR: AddressSanitizer: global-buffer-overflow on address 0x560e33c2993e
READ of size 16 at 0x560e33c2993e thread T0
#0 in __asan_memcpy
#1 in vReceiveRA /opt/oss-sources/freertos-tcp/source/FreeRTOS_RA.c:409:38
#2 in LLVMFuzzerTestOneInput fuzz_ra_ep.c:65:5
0x560e33c2993e is located 2 bytes to the right of global variable 'buf' of size 1500
SUMMARY: AddressSanitizer: global-buffer-overflow in __asan_memcpy
```

The 16-byte prefix copy at line 409 starts 2 bytes past the received data, because 6 of the 8 bytes between the option and the prefix field are still inside the frame. With the option as the very last bytes of the frame, the same copy starts 8 bytes past the end and the lifetime read at line 414 crosses it too, which gives the 24 bytes above. The harness buffer is a fixed array: on a device, whether the bytes past the frame are still inside the heap block depends on the buffer scheme described above.

04 Fix and recommendation

Commit `d210ba0` (main, released in V4.4.1; backported as `5f87b95` in V4.2.6 with identical source hunks) rejects a Prefix Information option shorter than the structure, marks the packet as malformed and stops parsing it:

- commit `d210ba0`, `source/FreeRTOS_RA.c`

```
@@ -276,6 +276,7 @@
const size_t uxLast = pxNetworkBuffer->xDataLength - uxNeededSize;
uint8_t * pucBytes = &( pxNetworkBuffer->pucEthernetBuffer[ uxNeededSize ] );
ICMPPrefixOption_IPv6_t * pxPrefixOption = NULL;
+ BaseType_t xMalformed = pdFALSE;
while( ( uxIndex + 1U ) < uxLast )
{
@@ -312,16 +313,29 @@
break;
case ndICMP_PREFIX_INFORMATION: /* 3 */
- /* MISRA Ref 11.3.1 [Misaligned access] */
- /* More details at: https://github.com/FreeRTOS/FreeRTOS-Plus-TCP/blob/main/MISRA.md#rule-113 */
- /* coverity[misra_c_2012_rule_11_3_violation] */
pxPrefixOption = ( ( ICMPPrefixOption_IPv6_t * ) &( pucBytes[ uxIndex ] ) );
- FreeRTOS_printf( ( "RA: Prefix len %d Life %u, %u (%p)\n",
- pxPrefixOption->ucPrefixLength,
- ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulValidLifeTime ),
- ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulPreferredLifeTime ),
- ( void * ) pxPrefixOption->ucPrefix ) );
+
+ if( uxLength < sizeof( ICMPPrefixOption_IPv6_t ) )
+ {
+ FreeRTOS_printf(
+ ( "RA: Prefix option too short ( %u < %u )\n",
+ ( unsigned ) uxLength,
+ ( unsigned ) sizeof( ICMPPrefixOption_IPv6_t ) );
+ xMalformed = pdTRUE;
+ }
+ else
+ {
+ /* MISRA Ref 11.3.1 [Misaligned access] */
+ /* More details at: https://github.com/FreeRTOS/FreeRTOS-Plus-TCP/blob/main/MISRA.md#rule-113 */
+ /* coverity[misra_c_2012_rule_11_3_violation] */
pxPrefixOption = ( ( ICMPPrefixOption_IPv6_t * ) &( pucBytes[ uxIndex ] ) );
+ FreeRTOS_printf( ( "RA: Prefix len %d Life %u, %u (%p)\n",
+ pxPrefixOption->ucPrefixLength,
+ ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulValidLifeTime ),
+ ( unsigned ) FreeRTOS_ntohl( pxPrefixOption->ulPreferredLifeTime ),
+ ( void * ) pxPrefixOption->ucPrefix ) );
+ }
break;
case ndICMP_REDIRECTED_HEADER: /* 4 */
@@ -345,6 +359,12 @@
break;
}
+ if( xMalformed != pdFALSE )
+ {
+ FreeRTOS_printf( ( "RA: Malformed packet.\n" ) );
+ break;
+ }
+
uxIndex = uxIndex + uxLength;
} /* while( ( uxIndex + 1 ) < uxLast ) */
```

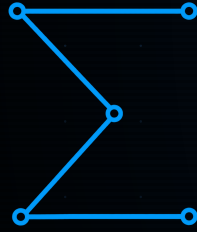
Upgrade to FreeRTOS-Plus-TCP V4.4.1 or V4.2.6, or later. Forks of the stack should port `d210ba0`. Where an upgrade is not possible, the vendor advisory suggests filtering untrusted Router Advertisements on the segment, or isolating devices where rogue RAs cannot be injected.

05 Timeline, credit, references

DATE (UTC)	EVENT
2026-04-13	Fix authored upstream: commit d210ba0 "Validate PREFIX_INFORMATION option size in RA"
2026-04-29	FreeRTOS-Plus-TCP V4.4.1 and V4.2.6 released with the fix
2026-04-29	CVE-2026-7425 reserved and published by Amazon (CNA); GHSA-gffr-xgjj-jh9j published

Reported to AWS Security under coordinated disclosure by Marouane El Moufid (Eun0us), Epsilon, credited in the GHSA. Our initial report date is not kept in our archive and is not listed.

Advisory	github.com/FreeRTOS/FreeRTOS-Plus-TCP/security/advisories/GHSA-gffr-xgjj-jh9j
CVE record	www.cve.org/CVERecord?id=CVE-2026-7425
Fix (main)	commit d210ba006ac2d5ee0815dc385bfecdcc6b566479
Fix (V4.2.x)	commit 5f87b9525c0bde4f4399fa7737285b0ab69701dc
Releases	V4.4.1, V4.2.6
AWS bulletin	aws.amazon.com/security/security-bulletins/2026-023-aws/



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 • France