



ESPILON
SECURITY RESEARCH

SECURITY ADVISORY

Out-of-bounds write via unsanitized prefix length in Router Advertisement processing

CVE-2026-7426

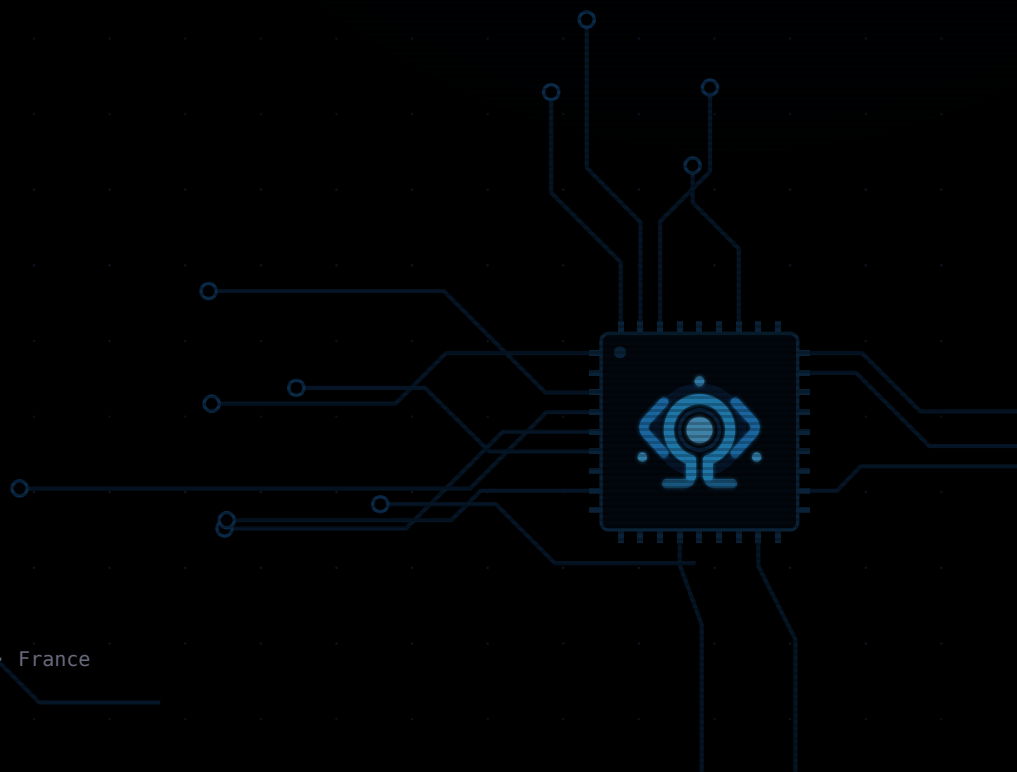
CVSS 8.1 · HIGH

AWS · FreeRTOS-Plus-TCP

CVE PUBLISHED 29 Apr 2026

Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France



01 Vulnerability

8.1

HIGH

CVSS 3.1 BASE SCORE · AMAZON (CNA)

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

CVSS 4.0 (CVE record): 6.1 MEDIUM · AV:A/AC:L/AT:P/PR:N/UI:N/VC:N/VI:H/VA:H/SC:N/SI:N/SA:N

CVE	CVE-2026-7426
Advisory	GHSA-97qg-4359-xm3x
Component	FreeRTOS-Plus-TCP, vReceiveRA() in source/FreeRTOS_RA.c and FreeRTOS_CreateIPv6Address() in source/FreeRTOS_ND.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 write (CWE-787)
Configuration	ipconfigUSE_IPv6 and ipconfigUSE_RA (both enabled by default), an end-point using RA/SLAAC (bWanRA)
Credit	Epsilon · Marouane El Moufid (Eun0us)

Summary

FreeRTOS-Plus-TCP configures an IPv6 address by SLAAC from the Prefix Information option of a Router Advertisement (RA). The 8-bit Prefix Length field of that option is stored in the end-point without any range check, and later used as a byte count when the stack builds its address. A value above 128 makes FreeRTOS_CreateIPv6Address() copy up to 32 bytes into a 16-byte IPv6 address, then modify one of those out-of-range bytes again. An unauthenticated attacker on the same link triggers it with one crafted RA while the device is waiting for a router.

This advisory is distinct from CVE-2026-7425 (GHSA-gffr-xgjj-jh9j), an out-of-bounds *read* in the same RA parser when the Prefix Information option is shorter than its structure (fix d210ba0). The defect here is the Prefix *Length* value and the *write* it drives (fix ee62f80).

02 Technical details

Root cause

When an RA with a non-zero router lifetime carries a Prefix Information option, vReceiveRA() copies the option into every end-point that is waiting for an RA. The Prefix Length goes straight from the wire into uxPrefixLength:

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

406 if( ( pxEndPoint->bits.bWanRA != ipFALSE_BOOL ) && ( pxEndPoint->xRADATA.eRAState == eRAStateWait ) )
407 {
408     pxEndPoint->ipv6_settings.uxPrefixLength = pxPrefixOption->ucPrefixLength;
409     ( void ) memcpy( pxEndPoint->ipv6_settings.xPrefix.ucBytes, pxPrefixOption->ucPrefix,
ipSIZE_OF_IPv6_ADDRESS );
410     ( void ) memcpy( pxEndPoint->ipv6_settings.xGatewayAddress.ucBytes, pxICMPPacket-
>xIPHeader.xSourceAddress.ucBytes, ipSIZE_OF_IPv6_ADDRESS );
```

Lines 412 to 418 then set the state to eRAStateIPTest and run vRAProcess(), which calls FreeRTOS_CreateIPv6Address() with that stored length (FreeRTOS_RA.c:580) to build a random address from the prefix. FreeRTOS_CreateIPv6Address() treats the length as trusted. Its only guard is a configASSERT:

• source/FreeRTOS_ND.c, FreeRTOS_CreateIPv6Address() (V4.4.0)

```
1355 configASSERT( ( uxPrefixLength > 0U ) && ( uxPrefixLength <= ( 8U * ipSIZE_OF_IPv6_ADDRESS ) ) );
1356
1357 if( uxPrefixLength >= 8U )
1358 {
1359     ( void ) memcpy( pxIPAddress->ucBytes, pxPrefix->ucBytes, ( uxPrefixLength + 7U ) / 8U );
1360 }
1361
1362 pucSource = ( uint8_t * ) pulRandom;
1363 uxIndex = uxPrefixLength / 8U;
1364
1365 if( ( uxPrefixLength % 8U ) != 0U )
1366 {
1367     /* uxHostLen is between 1 and 7 bits long. */
1368     size_t uxHostLen = 8U - ( uxPrefixLength % 8U );
1369     uint32_t uxHostMask = ( ( uint32_t ) 1U ) << uxHostLen - 1U;
1370     uint8_t ucNetMask = ( uint8_t ) ~( uxHostMask );
1371
1372     pxIPAddress->ucBytes[ uxIndex ] &= ucNetMask;
1373     pxIPAddress->ucBytes[ uxIndex ] |= ( pucSource[ 0 ] & ( ( uint8_t ) uxHostMask ) );
1374     pucSource = &( pucSource[ 1 ] );
1375     uxIndex++;
1376 }
1377
1378 if( uxIndex < ipSIZE_OF_IPv6_ADDRESS )
1379 {
1380     ( void ) memcpy( &( pxIPAddress->ucBytes[ uxIndex ] ), pucSource, ipSIZE_OF_IPv6_ADDRESS - uxIndex );
```

With a Prefix Length L between 129 and 255:

- line 1359 copies $(L + 7) / 8$ bytes, 17 to 32, into `pxIPAddress->ucBytes`, a 16-byte array, and reads the same count from the 16-byte `xPrefix`;
- line 1363 sets `uxIndex = L / 8`, 16 to 31; when $L \% 8 \neq 0$, lines 1372 and 1373 then modify `ucBytes[uxIndex]`, which is past the array as well (it is the last byte the `memcpy` wrote);
- the final `memcpy` at line 1380 is skipped because `uxIndex` is no longer below 16.

What is overwritten

The destination is `ipv6_settings.xIPAddress`, the first field of `IPV6Parameters_t` (`FreeRTOS_Routing.h:140`). In the same structure it is followed by `uxPrefixLength` (a `size_t`), then `xPrefix`, `xGatewayAddress` and the DNS server addresses. So the overflow reaches `uxPrefixLength` and the first bytes of `xPrefix`. The source of the copy is `xPrefix` followed by `xGatewayAddress`, which `vReceiveRA()` filled from the attacker's prefix and from the RA's IPv6 source address, so the written bytes are attacker-chosen. For the larger values (from 161 on a 32-bit target, where `xPrefix` starts 20 bytes after `xIPAddress`, so that more than 20 bytes are copied) the source and destination ranges overlap, which is undefined behaviour for `memcpy`. With a maximum of 32 bytes from the start of `xIPAddress`, the write stays inside the end-point object; it corrupts the end-point's own addressing state, including the stored prefix length itself.

Attacker position and preconditions

- Adjacent network: the attacker sends one ICMPv6 RA (type 134) on the device's link. No authentication, no prior exchange.
- `ipconfigUSE_IPv6` and `ipconfigUSE_RA` compiled in (both are enabled by default in `FreeRTOSIPConfigDefaults.h`), and an end-point the application configured for SLAAC (`bWantRA`).
- The end-point is in `eRAStateWait`, waiting for an answer to its Router Solicitation: at start-up and again each time its RA lease expires. The solicitation is multicast, visible on the link.
- Builds that define `configASSERT` stop at line 1355 instead: the device halts, a denial of service. Builds where `configASSERT` expands to nothing reach the out-of-bounds write.

03 Impact and evidence

Impact. The established effect is a bounded out-of-bounds write of attacker-chosen bytes (at most 16 bytes past the 16-byte address array) into the end-point structure, which corrupts the device's IPv6 addressing state, or an assertion halt where `configASSERT` is enabled. The CNA rates integrity and availability High and confidentiality None. We have not demonstrated control-flow hijack or code execution and make no such claim.

Evidence. No sanitizer trace for this finding is kept in our archive; the evidence is the code above and the vendor's fix below, which adds the missing range check at both places. From the code, the trigger is an RA with a valid ICMPv6 checksum, a non-zero router lifetime (`FreeRTOS_RA.c:392`) and a full-size Prefix Information option whose Prefix Length is 129 to 255, received in `eRAStateWait`. The write stays inside one object, so `AddressSanitizer` would not flag it in a typical build.

04 Fix and recommendation

Commit ee62f80 (main, released in V4.4.1; backported as c6f4a1b in V4.2.6 with the same added and removed lines) rejects a Prefix Length of 0 or above 128 when the RA is received, and makes FreeRTOS_CreateIPv6Address() fail on such a length and bound the partial-byte index:

• commit ee62f80, source/FreeRTOS_RA.c

```
@@ -425,6 +425,14 @@
{
    if( ( pxEndPoint->bits.bWanRA != ipFALSE_BOOL ) && ( pxEndPoint->xRData.eRAState == eRAStateWait ) )
    {
+       if( ( pxPrefixOption->ucPrefixLength == 0U ) ||
+         ( pxPrefixOption->ucPrefixLength > ( 8U * ipSIZE_OF_IPv6_ADDRESS ) ) )
+       {
+           FreeRTOS_printf( ( "vReceiveRA: The prefix length "
+                             "is invalid\n" ) );
+           break;
+       }
        pxEndPoint->ipv6_settings.ucPrefixLength = pxPrefixOption->ucPrefixLength;
        ( void ) memcpy( pxEndPoint->ipv6_settings.xPrefix.ucBytes, pxPrefixOption->ucPrefix, ipSIZE_OF_IPv6_ADDRESS );
        ( void ) memcpy( pxEndPoint->ipv6_settings.xGatewayAddress.ucBytes, pxICMPPacket->xIPHeader.xSourceAddress.ucBytes,
ipSIZE_OF_IPv6_ADDRESS );
    }
```

• commit ee62f80, source/FreeRTOS_ND.c (first 14 lines of the 68-line hunk)

```
@@ -1361,30 +1361,51 @@
/* A loopback IP-address has a prefix of 128. */
configASSERT( ( uxPrefixLength > 0U ) && ( uxPrefixLength <= ( 8U * ipSIZE_OF_IPv6_ADDRESS ) ) );
- if( uxPrefixLength >= 8U )
+ if( ( uxPrefixLength == 0U ) || ( uxPrefixLength > ( 8U * ipSIZE_OF_IPv6_ADDRESS ) ) )
+ {
+     FreeRTOS_printf( ( "Invalid prefix length %u\n",
+                       ( unsigned ) uxPrefixLength ) );
+     xResult = pdFAIL;
+ }
+ else if( uxPrefixLength >= 8U )
+ {
+     ( void ) memcpy( pxIPAddress->ucBytes, pxPrefix->ucBytes, ( uxPrefixLength + 7U ) / 8U );
+ }
```

The remaining 54 lines move the partial-byte write and the final memcpy under if(xResult == pdPASS) and add uxIndex < ipSIZE_OF_IPv6_ADDRESS to the partial-byte condition.

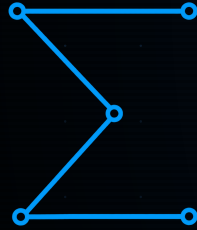
Upgrade to FreeRTOS-Plus-TCP V4.4.1 or V4.2.6, or later. Forks of the stack should take both files of ee62f80: the check in vReceiveRA() alone leaves FreeRTOS_CreateIPv6Address() open to any other caller. Where an upgrade is not possible, the vendor advisory suggests filtering untrusted Router Advertisements 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 ee62f80 "Validate RA prefix length bounds"
2026-04-29	FreeRTOS-Plus-TCP V4.4.1 and V4.2.6 released with the fix
2026-04-29	CVE-2026-7426 reserved and published by Amazon (CNA); GHSA-97qg-4359-xm3x published

Reported to AWS Security under coordinated disclosure by Marouane El Moufid (Eun0us), Espilon, 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-97qg-4359-xm3x
CVE record	www.cve.org/CVERecord?id=CVE-2026-7426
Fix (main)	commit ee62f8006a98fb89e9dec3c4ca5fc3cf5b3662e7
Fix (V4.2.x)	commit c6f4a1bb67713ea099a61b757d4e8716bd2dae60
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