



ESPILON
SECURITY RESEARCH

SECURITY ADVISORY

Integer underflow in the DHCPv6 sub-option parser

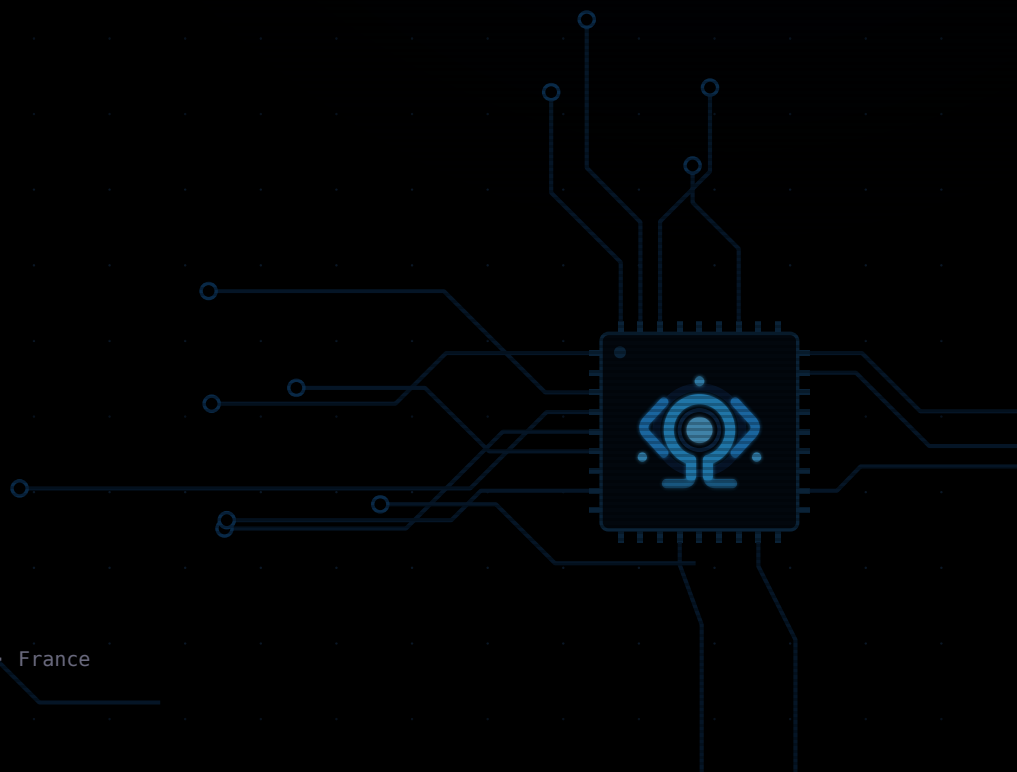
CVE-2026-7424 **CVSS 8.1 · HIGH**

AWS · FreeRTOS-Plus-TCP

CVE PUBLISHED **29 Apr 2026**

Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France



01 Vulnerability

8.1

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): 7.2 HIGH · AV:A/AC:L/AT:N/PR:N/UI:N/VC:N/VI:H/VA:H/SC:N/SI:N/SA:N

HIGH

CVE	CVE-2026-7424
Advisory	GHSA-wrhm-c99p-2p8g
Component	FreeRTOS-Plus-TCP, prvDHCPv6_subOption() in source/FreeRTOS_DHCPv6.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	Integer underflow (CWE-191)
Configuration	ipconfigUSE_IPv6 and ipconfigUSE_DHCPv6 (the latter disabled by default), an IPv6 end-point using DHCPv6 (bWantDHCP)
Credit	Epsilon · Marouane El Moufid (EunOus)

Summary

The DHCPv6 client in FreeRTOS-Plus-TCP parses the sub-options of the IA_NA and IA_PD options in prvDHCPv6_subOption(). An unknown sub-option is skipped with a byte count computed as uxLength2 - uxUsed: the sub-option's own length minus the number of bytes already consumed in the parent option. When the sub-option is shorter than that count, the unsigned subtraction wraps. The bounds check of the reader wraps with it and passes, and the read index moves *backwards*. The parser then re-reads bytes it has already consumed. One crafted DHCPv6 reply from an adjacent attacker is enough; the vendor states that it can corrupt the device's IPv6 address assignment, DNS configuration and lease times, and freeze the IP task permanently.

02 Technical details

Root cause

uxUsed counts bytes from pxSet->uxStart, the first byte after the parent option's code and length (FreeRTOS_DHCPv6.c:1498). It is first computed at line 1127, after the 12 bytes of IAID, T1 and T2 have been read. Each loop iteration then reads a sub-option header, recomputes uxUsed (so it is at least 16 for the first sub-option) and validates the length:

• source/FreeRTOS_DHCPv6.c, prvDHCPv6_subOption() (V4.4.0)

```
1160 usOption2 = usBitConfig_read_16( pxMessage );
1161 uxLength2 = usBitConfig_read_16( pxMessage );
1162
1163 uxUsed = pxMessage->uxIndex - pxSet->uxStart;
1164
1165 /* Check sub-option length. */
1166 if( prvIsOptionLengthValid( usOption2, uxLength2, pxMessage->uxSize - pxMessage->uxIndex ) != pdTRUE )
1167 {
1168     FreeRTOS_printf( ( "prvDHCPv6_subOption: %u has invalid length %u, remaining buffer size %u\n",
1169                       usOption2,
1170                       ( unsigned ) uxLength2,
1171                       ( unsigned ) ( pxMessage->uxSize - pxMessage->uxIndex ) ) );
1172     xReturn = pdFALSE;
1173     break;
1174 }
```

prvIsOptionLengthValid() has no minimum for codes it does not know (uxMinOptLength = 0U, line 221) and only checks that the length fits in the remaining buffer, so any small uxLength2 passes. The unknown code then lands in the default case:

• source/FreeRTOS_DHCPv6.c, prvDHCPv6_subOption() (V4.4.0)

```
1198     default:
1199         ( void ) xBitConfig_read_uc( pxMessage, NULL, uxLength2 - uxUsed );
1200         FreeRTOS_printf( ( "prvDHCPv6Analyse: skipped unknown option %u\n", usOption2 ) );
1201         break;
1202     }
1203
1204     if( xReturn != pdTRUE )
1205     {
1206         FreeRTOS_printf( ( "prvDHCPv6_subOption: One of sub-options %d returns error\n", usOption2 ) );
1207         break;
1208     }
1209
1210     /* Update remaining length. */
1211     uxUsed = pxMessage->uxIndex - pxSet->uxStart;
1212     uxRemain = 0U;
1213
1214     if( pxSet->uxOptionLength > uxUsed )
1215     {
1216         uxRemain = pxSet->uxOptionLength - uxUsed;
1217     }
1218 }
```

uxLength2 is a uint16_t and uxUsed a size_t; when uxLength2 < uxUsed the difference is a value close to SIZE_MAX. It is passed to the reader as a skip count:

• source/FreeRTOS_BitConfig.c, xBitConfig_read_uc() (V4.4.0)

```
106 if( pxConfig->xHasError == pdFALSE )
107 {
108     if( ( pxConfig->uxIndex + uxNeeded ) <= pxConfig->uxSize )
109     {
110         if( pucData != NULL )
111         {
112             ( void ) memcpy( pucData, &( pxConfig->ucContents[ pxConfig->uxIndex ] ), uxNeeded );
113         }
114         else
115         {
116             /* Caller just wants to skip some bytes. */
117         }
118
119         pxConfig->uxIndex += uxNeeded;
120         xResult = pdTRUE;
121     }
}
```

The addition at line 108 wraps as well, so the check passes, and line 119 performs the same wrapped addition on the index. Since uxUsed = uxIndex - uxStart, the result is uxIndex = uxStart + uxLength2: the index is moved back inside the parent option, to an offset the attacker picks with the 16-bit length field.

Consequences in the parser

The function returns to its loop with a valid-looking index that points at bytes already parsed. The next iterations re-read them as sub-option headers, so attacker-placed bytes are parsed under a different interpretation (for example as an IA Address or IA Prefix sub-option, which write the address, prefix and lifetimes into the DHCPv6 message). From the code, the freeze follows directly: with uxLength2 = 12 the index returns to the first sub-option header itself, uxRemain stays positive, and the loop reads and skips the same header forever inside the IP task. The vendor advisory describes the outcome as a permanent IP task freeze requiring a hardware reset.

Attacker position and preconditions

- Adjacent network, pre-authentication: the attacker answers on the link with a DHCPv6 message of an accepted type (Advertise, Confirm, Reply or Decline, lines 1461 to 1465).
- The reply must carry the 24-bit transaction ID of the client's current request (FreeRTOS_DHCPv6.c:1445). The client sends its Solicit and Request to the multicast address ff02::1:2, so the ID is visible to anyone on the link.
- ipconfigUSE_IPv6 and ipconfigUSE_DHCPv6 compiled in, and an IPv6 end-point configured for DHCPv6. ipconfigUSE_DHCPv6 is disabled in FreeRTOSIPConfigDefaults.h, so only products that enable the DHCPv6 client are exposed.

03 Impact and evidence

Impact. Established from the code: a wrapped skip count that rewinds the parser and lets one packet be parsed twice, and an endless loop in the IP task for a chosen length value, which stops all networking on the device until reset. The CNA rates integrity and availability High and confidentiality None. From the code, the rewound index stays inside

the received message, so this defect does not by itself read or write outside the receive buffer; we make no code-execution claim.

Evidence. Our fuzzing notes record this finding as DHCPV6-001, found with a libFuzzer harness that feeds `vDHCPv6Process()` on the FreeRTOS-Plus-TCP tree of 2026-03-16 (af6b379). The campaign kept its hang inputs. We replayed one of them, 218 bytes written on 2026-03-31, on 2026-10-28 with the campaign's binary. libFuzzer stops it after the 3-second limit inside the sub-option loop: `usBitConfig_read_16()` $\leftarrow$ `prvDHCPv6_subOption()` (`FreeRTOS_DHCPv6.c:1160`) $\leftarrow$ `prvDHCPv6_handleOption()` $\leftarrow$ `prvDHCPv6Analyse()` $\leftarrow$ `vDHCPv6Process()`. Under a debugger, the skip at line 1199 reaches `xBitConfig_read_uc()` with a size of `0xffffffffffffff2` at index 30, then `0xffffffffffffffc` at index 20, and the second call repeats with the same values. As described above, the skip wraps the index back to byte 16 (lines 108 and 119), so the same sub-option header is read again on every pass. The full log is kept with this report.

04 Fix and recommendation

Commit 0015547 (main, released in V4.4.1; backported as ca65d44 in V4.2.6 with identical source hunks) skips `uxLength2` bytes, the sub-option payload, instead of the wrapped difference, and leaves the loop when the reader has flagged an error:

```
• commit 0015547, source/FreeRTOS_DHCPv6.c

@@ -1196,13 +1196,14 @@ static BaseType_t prvDHCPv6_subOption( uint16_t usOption,
    break;
    default:
-    ( void ) xBitConfig_read_uc( pxMessage, NULL, uxLength2 - uxUsed );
+    ( void ) xBitConfig_read_uc( pxMessage, NULL, uxLength2 );
    FreeRTOS_printf( ( "prvDHCPv6Analyse: skipped unknown option %u\n", usOption2 ) );
    break;
}
- if( xReturn != pdTRUE )
+ if( ( xReturn != pdTRUE ) || ( pxMessage->xHasError != pdFALSE ) )
{
+    xReturn = pdFALSE;
    FreeRTOS_printf( ( "prvDHCPv6_subOption: One of sub-options %d returns error\n", usOption2 ) );
    break;
}
```

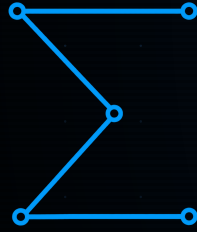
Upgrade to FreeRTOS-Plus-TCP V4.4.1 or V4.2.6, or later. Forks of the stack should port 0015547. The vendor advisory suggests, where an upgrade is not possible, disabling DHCPv6 when addresses can be configured statically, or filtering DHCPv6 traffic to trusted sources on the segment.

05 Timeline, credit, references

DATE (UTC)	EVENT
2026-04-08	Our follow-up to AWS Security on this report: no patch yet for this issue
2026-04-13	Fix authored upstream: commit 0015547 "Fix DHCPv6 sub-option default case skip"
2026-04-29	FreeRTOS-Plus-TCP V4.4.1 and V4.2.6 released with the fix
2026-04-29	CVE-2026-7424 reserved and published by Amazon (CNA); GHSA-wrhm-c99p-2p8g published

Reported to AWS Security under coordinated disclosure by Marouane El Moufid (Eun0us), Epsilon, credited in the GHSA. The initial report predates our 2026-04-08 follow-up; its exact date is not kept in our archive.

Advisory	github.com/FreeRTOS/FreeRTOS-Plus-TCP/security/advisories/GHSA-wrhm-c99p-2p8g
CVE record	www.cve.org/CVERecord?id=CVE-2026-7424
Fix (main)	commit 00155471eafd05160ba8ce352deaa8e51d65e05f
Fix (V4.2.x)	commit ca65d44e7434267eff7a95d0d2e9c6ab63ad18dd
Releases	V4.4.1, V4.2.6
AWS bulletin	aws.amazon.com/security/security-bulletins/2026-022-aws/



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 • France