ESPILON
SECURITY RESEARCH

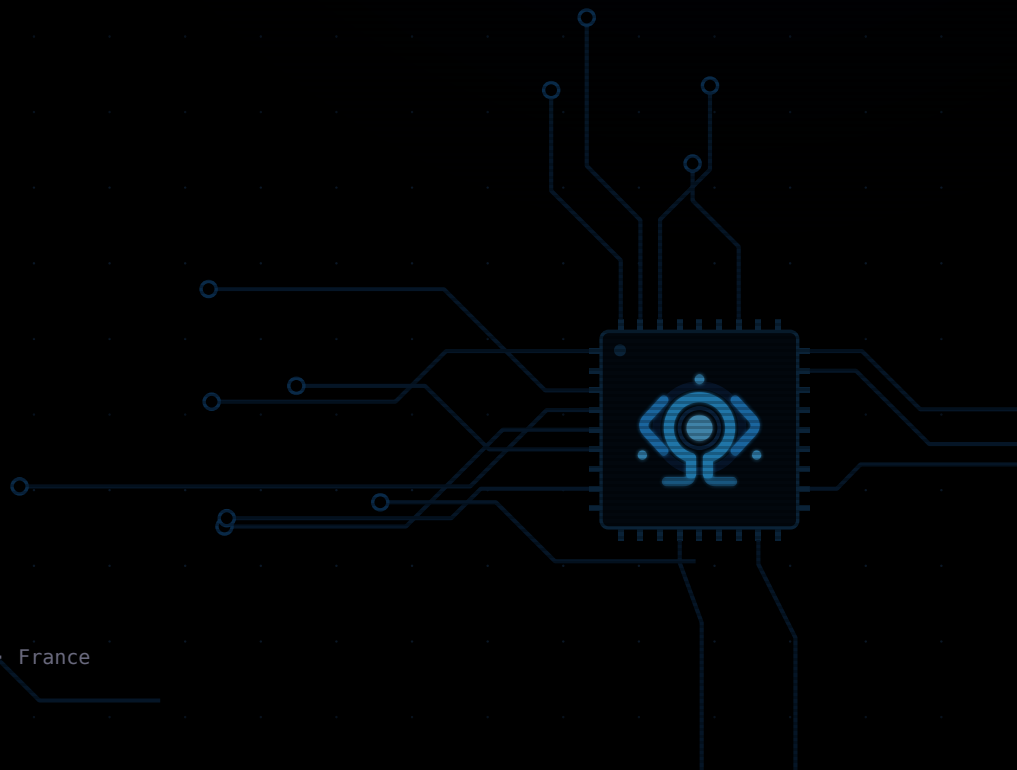
SECURITY ADVISORY

MAC address validation bypass in IPv4/IPv6 packet processing

CVE-2026-7422 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 MEDIUM

CVSS 3.1 BASE SCORE · AMAZON (CNA)

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

CVSS 4.0 (CVE record): 7.1 HIGH · AV:A/AC:L/AT:N/PR:N/UI:N/VC:N/VI:H/VA:N/SC:N/SI:N/SA:N

CVE	CVE-2026-7422
Advisory	GHSA-jpw4-6h59-62w9
Component	FreeRTOS-Plus-TCP, `prvAllowIPPacketIPv4()` in `source/FreeRTOS_IPv4.c` and `prvAllowIPPacketIPv6()` in `source/FreeRTOS_IPv6.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	Authentication bypass by spoofing (CWE-290)
Configuration	Software checksum validation, `ipconfigDRIVER_INCLUDED_RX_IP_CHECKSUM == 0` (the default)
Credit	Espilon · Marouane El Moufid (Eun0us)

Summary

Before an incoming IPv4 or IPv6 packet is handed to the protocol handlers, FreeRTOS-Plus-TCP verifies its checksums, and the same routine performs the minimum-size checks of the IP and transport headers. Both receive paths skip that routine when the Ethernet *source* MAC address of the frame equals the MAC address of one of the device's own end-points, on the assumption that such a frame is local loopback. The source MAC is set by the sender. An adjacent attacker who puts the device's own MAC address in the source field gets frames with wrong checksums and undersized headers accepted and processed.

02 Technical details

Root cause

In the default configuration the stack validates received packets in software. The IPv4 path looks up an end-point by the frame's source MAC and only validates when none is found:

source/FreeRTOS_IPv4.c, prvAllowIPPacketIPv4() (V4.4.0)

```
484 #if ( ipconfigDRIVER_INCLUDED_RX_IP_CHECKSUM == 0 )
485 {
486     /* Some drivers of NIC's with checksum-offloading will enable the above
487      * define, so that the checksum won't be checked again here */
488     if( eReturn == eProcessBuffer )
489     {
490         const NetworkEndPoint_t * pxEndPoint = FreeRTOS_FindEndPointOnMAC( &( pxIPPacket-
>xEthernetHeader.xSourceAddress ), NULL );
491
492         /* Do not check the checksum of loop-back messages. */
493         if( pxEndPoint == NULL )
494         {
495             /* Is the IP header checksum correct?
```

`FreeRTOS_FindEndPointOnMAC()` with a `NULL` interface returns any end-point whose MAC address equals the one given (`FreeRTOS_Routing.c:432`). Nothing ties a frame carrying that source address to the device itself: it is a field of the received frame. Everything inside the `if` at line 493, lines 495 to 518, is then skipped: the IPv4 header checksum at line 503 and the protocol checksum at line 509. The IPv6 path has the same structure:

source/FreeRTOS_IPv6.c, prvAllowIPPacketIPv6() (V4.4.0)

```
543 if( eReturn == eProcessBuffer )
544 {
545     /* MISRA Ref 11.3.1 [Misaligned access] */
546     /* More details at: https://github.com/FreeRTOS/FreeRTOS-Plus-TCP/blob/main/MISRA.md#rule-113 */
547     /* coverity[misra_c_2012_rule_11_3_violation] */
548     const IPPacket_t * pxIPPacket = ( ( const IPPacket_t * ) pxNetworkBuffer->pucEthernetBuffer );
549     const NetworkEndPoint_t * pxEndPoint = FreeRTOS_FindEndPointOnMAC( &( pxIPPacket-
    >xEthernetHeader.xSourceAddress ), NULL );
550
551     /* IPv6 does not have a separate checksum in the IP-header */
552     /* Is the upper-layer checksum (TCP/UDP/ICMP) correct? */
553     /* Do not check the checksum of loop-back messages. */
554     if( pxEndPoint == NULL )
555     {
556         if( usGenerateProtocolChecksum( ( uint8_t * ) ( pxNetworkBuffer->pucEthernetBuffer ), pxNetworkBuffer-
    >xDataLength, pdFALSE ) != ipCORRECT_CRC )
557         {
558             /* Protocol checksum not accepted. */
559             eReturn = eReleaseBuffer;
560         }
561     }
```

What is skipped

`usGenerateProtocolChecksum()` is more than a checksum. Before computing the checksum, it runs `prvChecksumIPv4Checks()` or `prvChecksumIPv6Checks()` (`FreeRTOS_IP_Utils.c:1123` and `1135`) and then `prvChecksumProtocolChecks()` (line 1155), which check that the buffer holds the IP header, the extension headers and the transport header the packet claims. The `#else` branch of the same functions, used when the network driver validates checksums, calls `xCheckIPv4SizeFields()` and `xCheckIPv6SizeFields()` instead (`FreeRTOS_IPv4.c:525`, `FreeRTOS_IPv6.c:568`). On the software path, a spoofed source MAC therefore removes, in one step:

- the IPv4 header checksum check;
- the TCP, UDP and ICMP checksum checks, for IPv4 and IPv6;
- the minimum-size checks that live in the checksum routine.

Attacker position and preconditions

- Adjacent network: the attacker sends Ethernet frames on the device's segment whose source MAC is the device's own MAC address, which it can read from any frame the device transmits. No authentication.
- `ipconfigDRIVER_INCLUDED_RX_IP_CHECKSUM` is 0, the default in `FreeRTOSIPConfigDefaults.h`. Builds that set it to 1 use the `#else` branch shown above and do not contain this bypass.
- The frame must still pass the address checks that precede this block (destination MAC and IP address accepted by the end-point).

03 Impact and evidence

Impact. Established from the code: frames with invalid IPv4 header checksums, invalid TCP, UDP or ICMP checksums, or truncated headers are accepted and passed to the protocol handlers, which then rely on validation that did not run. The vendor advisory states that the bypass skips all checksum and minimum-size validation. The CNA rates integrity High and confidentiality and availability None. We make no claim about what a particular handler does with an undersized packet.

Evidence. No packet capture or sanitizer trace for this finding is kept in our archive. The evidence is the code above and the vendor's fix, which removes the MAC-based exception in both files.

04 Fix and recommendation

Commit `87781fa` (main, released in V4.4.1; backported as `2c295c3` in V4.2.6 with identical source hunks) removes the source-MAC lookup so that checksums, and the size checks inside them, always run:

commit 87781fa, source/FreeRTOS_IPv6.c

```
@@ -125,7 +125,7 @@ const struct xIPv6_Address FreeRTOS_in6addr_loopback = { { 0U, 0U, 0U, 0U, 0U, 0
 /* Check if the complete IPv6-header plus protocol data have been transferred: */
 usPayloadLength = FreeRTOS_ntohs( pxIPv6Packet->xIPHeader.usPayloadLength );
-if( uxBufferLength != ( size_t ) ( ipSIZE_OF_ETH_HEADER + ipSIZE_OF_IPv6_HEADER + ( size_t ) usPayloadLength ) )
+if( uxBufferLength < ( size_t ) ( ipSIZE_OF_ETH_HEADER + ipSIZE_OF_IPv6_HEADER + ( size_t ) usPayloadLength ) )
 {
     DEBUG_SET_TRACE_VARIABLE( xLocation, 4 );
     break;
@@ -542,22 +542,12 @@ eFrameProcessingResult_t prvAllowIPPacketIPv6( const IPHeader_IPv6_t * const pxI
      * define, so that the checksum won't be checked again here */
     if( eReturn == eProcessBuffer )
     {
-        /* 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] */
-        const IPPacket_t * pxIPPacket = ( ( const IPPacket_t * ) pxNetworkBuffer->pucEthernetBuffer );
-        const NetworkEndPoint_t * pxEndPoint = FreeRTOS_FindEndPointOnMAC( &( pxIPPacket->xEthernetHeader.xSourceAddress ), NULL );
-
         /* IPv6 does not have a separate checksum in the IP-header */
         /* Is the upper-layer checksum (TCP/UDP/ICMP) correct? */
-        /* Do not check the checksum of loop-back messages. */
-        if( pxEndPoint == NULL )
+        if( usGenerateProtocolChecksum( ( uint8_t * ) ( pxNetworkBuffer->pucEthernetBuffer ), pxNetworkBuffer->xDataLength, pdFALSE ) !=
ipCORRECT_CRC )
         {
-            if( usGenerateProtocolChecksum( ( uint8_t * ) ( pxNetworkBuffer->pucEthernetBuffer ), pxNetworkBuffer->xDataLength, pdFALSE ) !=
ipCORRECT_CRC )
-            {
-                /* Protocol checksum not accepted. */
-                eReturn = eReleaseBuffer;
-            }
+            /* Protocol checksum not accepted. */
+            eReturn = eReleaseBuffer;
         }
     }
 }
```

commit 87781fa, source/FreeRTOS_IPv4.c (first 8 lines of the 55-line hunk)

```
@@ -487,34 +487,28 @@ enum eFrameProcessingResult prvAllowIPPacketIPv4( const struct xIP_PACKET * cons
     * define, so that the checksum won't be checked again here */
     if( eReturn == eProcessBuffer )
     {
-        const NetworkEndPoint_t * pxEndPoint = FreeRTOS_FindEndPointOnMAC( &( pxIPPacket->xEthernetHeader.xSourceAddress ), NULL );
-
-        /* Do not check the checksum of loop-back messages. */
-        if( pxEndPoint == NULL )
+        /* Is the IP header checksum correct?
```

The remaining 47 lines remove the braces of the `if( pxEndPoint == NULL )` block and re-indent the unchanged checksum checks by one level.

The first IPv6 hunk, in `xCheckIPv6SizeFields()` (compiled only when `ipconfigDRIVER_INCLUDED_RX_IP_CHECKSUM` is 1), is a separate change carried by the same commit: it accepts a buffer longer than the IPv6 payload length (for example a padded Ethernet frame) instead of requiring an exact match.

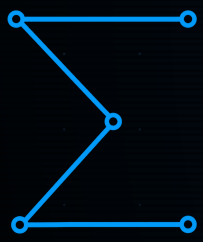
Upgrade to FreeRTOS-Plus-TCP V4.4.1 or V4.2.6, or later. Forks of the stack should port `87781fa`. Code derived from FreeRTOS-Plus-TCP that needs a loopback shortcut should base it on the buffer's origin inside the stack, never on a field of the received frame.

05 Timeline, credit, references

DATE (UTC)	EVENT
2026-03-31	Fix authored upstream: commit `87781fa` "Remove MAC-based loopback validation bypass"
2026-04-29	FreeRTOS-Plus-TCP V4.4.1 and V4.2.6 released with the fix
2026-04-29	CVE-2026-7422 reserved and published by Amazon (CNA); GHSA-jpw4-6h59-62w9 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-jpw4-6h59-62w9
CVE record	www.cve.org/CVERecord?id=CVE-2026-7422
Fix (main)	commit 87781fac8da1d400c39c82754748905d1d92fbb6
Fix (V4.2.x)	commit 2c295c34af63cf529b15ca49287e2a0786ed1df0
Releases	V4.4.1, V4.2.6
AWS bulletin	aws.amazon.com/security/security-bulletins/2026-021-aws/



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 · France