

CVE-2026-98592

Microsoft net: spacemit: clear TX descriptor on fragment mapping failure

In the Linux kernel, the following vulnerability has been resolved: net: spacemit: clear TX descriptor on fragment mapping failure
emac_tx_mem_map() writes TX_DESC_0_OWN into the ring descriptor for every slot beyond old_head as soon as that slot's memset()'d local copy is committed with `"*tx_desc_addr = tx_desc"`, i.e. before the buffers for that slot have necessarily all been mapped successfully.

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 9th percentile · FIRST	CISA DEADLINE — No CISA deadline
---	-------------------------	--	--

Signals behind the score

☒ counted in the score ☐ not counted

evidence for 2 of 8 signals

☐ CISA KEV	Not currently CISA KEV-listed	☐ Vendor exploitation	Active-exploitation status not yet checked - scored at the population base rate (none) until known
☐ Ransomware	Ransomware association unavailable	☐ KEV recency	CISA KEV listing date unavailable
☐ EPSS	0.002 probability	☐ Pre-patch disclosure	Disclosure status unavailable
☐ CVSS	CVSS base score unavailable	☐ Microsoft Exploitability Index	Not a Microsoft CVE

Remediation

Monitor Untagged reference available; patch status is not confirmed

NVD lists one or more references for this CVE that aren't tagged as a patch or vendor advisory. Review them directly to confirm patch availability and applicability before making a remediation decision.

Basis: NVD reference (untagged)

If `emac_tx_map_frag()` then fails on a later fragment, the `err_free_skb` path calls `emac_free_tx_buf()` to unmap and drop the skb, but leaves the already-written descriptor memory untouched, and `tx_ring->head` is never advanced past `old_head` (the `"tx_ring->head = head"` store is skipped by the goto). So a slot between `old_head` and the rolled-back head can be left with `TX_DESC_0_OWN` set and `buffer_addr_{1,2}` pointing at DMA mappings that `emac_free_tx_buf()` just tore down, while software considers that slot free again. The next successful `emac_tx_mem_map()` call only rebuilds `old_head` itself; if the DMA engine auto-advances into the following descriptor once it finishes `old_head`'s packet, it will fetch that stale, already-unmapped address. `emac_tx_clean_desc()` already treats `emac_free_tx_buf()` and clearing the descriptor as a pair when reclaiming completed descriptors; do the same in the mapping failure path.

Affected and fixed versions

PACKAGE	AFFECTED	FIXED IN	SOURCE
Not identified by the connected sources	—	no fixed version listed	NVD

Sources

Reference	git.kernel.org/stable/c/2d14720beb58870b15a52b236c3ab0be0e06e915	Additional reference	git.kernel.org/stable/c/3001a103a367c6051c43aff363b6979c10e65347
Additional reference	git.kernel.org/stable/c/ca9986cc824f2feb79b4a83e5f6f4c24c2140448	NVD	nvd.nist.gov/vuln/detail/CVE-2026-98592
ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-103637	Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98592

Snapshot at generation time. EPSS can move quickly. A public exploit module is not evidence of active exploitation.