

CVE-2026-98372

Microsoft xfrm: iptfs: fix stack OOB read in iptfs_skb_reset_frag_walk()

In the Linux kernel, the following vulnerability has been resolved: xfrm: iptfs: fix stack OOB read in iptfs_skb_reset_frag_walk() iptfs_skb_reset_frag_walk() advances to the fragment containing @offset with an unbounded loop: while (offset >= walk->past + walk->frags[walk->fragi].len) walk->past += walk->frags[walk->fragi++].len; walk->fragi is advanced and walk->frags[walk->fragi] is dereferenced without ever checking fragi against walk->nr_frags.

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 4.9th 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)

When the requested offset is at or beyond the total length spanned by the walk's fragments, fragi runs past nr_frags and off the end of the fixed-size on-stack frags[MAX_SKB_FRAGS + 1] array, reading out-of-bounds stack memory. The two callers behave differently: iptfs_skb_add_frags() already guards against this with if (!walk->nr_frags || offset >= walk->total + walk->initial_offset) return len; but iptfs_skb_can_add_frags() has no such guard and calls iptfs_skb_reset_frag_walk() unconditionally, so it performs the out-of-range walk. Its own "fragi < walk->nr_frags" bound check runs only afterwards, too late to prevent the read. This is reachable from the receive path: a crafted IP-TFS (AGGFRAG) payload delivered to an IPTFS SA drives iptfs_reassem_cont() -> iptfs_skb_can_add_frags() with an offset past the fragment total, e.g.: BUG: KASAN: stack-out-of-bounds in iptfs_skb_reset_frag_walk+0x235/0x250 Read of size 4 at addr ffff888008ad7210 by task repro/345 iptfs_skb_reset_frag_walk+0x235/0x250 net/xfrm/xfrm_iptfs.c:392 iptfs_skb_can_add_frags+0x155/0x310 net/xfrm/xfrm_iptfs.c:420 iptfs_reassem_cont+0xcf8/0x1140 net/xfrm/xfrm_iptfs.c:902 iptfs_input_ordered+0x552/0x670 net/xfrm/xfrm_iptfs.c:1280 iptfs_input+0x3d6/0xde0 net/xfrm/xfrm_iptfs.c:1741 xfrm_input+0x282f/0x6140 net/xfrm/xfrm_input.c:700 xfrm4_esp_rcv+0x93/0x120 net/ipv4/xfrm4_protocol.c:104 ip_rcv+0x278/0x2d0 net/ipv4/ip_input.c:612 Give iptfs_skb_can_add_frags() the same up-front guard that iptfs_skb_add_frags() already has, so the walk is never entered with an out-of-range offset. When it triggers, the caller falls back to the existing linearize-and-copy path, which is safe.

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/7e1c6884a0b494b7e3f204877f94c1e07a117bf7	Additional reference	git.kernel.org/stable/c/d042487dc118e494db2e2c1382310255c90ff544
Additional reference	git.kernel.org/stable/c/da56d0ee93d1bad779b0486f2fab92d1bc1f0cf3	NVD	nvd.nist.gov/vuln/detail/CVE-2026-98372
ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-93357	Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98372

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