

CVE-2026-98595

Microsoft udp: remove a disconnected socket from the 4-tuple hash table

In the Linux kernel, the following vulnerability has been resolved: udp: remove a disconnected socket from the 4-tuple hash table A UDP socket bound to a specific address and port keeps its entry in the 4-tuple hash table after it is disconnected: sk binds to 127.0.0.1:21001 sk connects to 127.0.0.2:20001 // filed in the 4-tuple table sk disconnects, connect(AF_UNSPEC) // still filed, peer now 0.0.0.0:0 __udp_disconnect() takes a socket out of that table only as a side effect of ->rehash() or ->unhash(), and it

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)

skips ->rehash() when SOCK_BINDADDR_LOCK is set and ->unhash() when SOCK_BINDPORT_LOCK is set. commit 6996a2d2d0a6 ("udp: Unhash auto-bound connected sk from 4-tuple hash table when disconnected.") fixed the same end state for a wildcard-bound socket, by a path this one does not take. The entry is counted whether or not anything hits it. hash4_cnt on the hash2 slot stays raised for as long as the socket lives, so udp_has_hash4() keeps sending every packet for that address and port through the 4-tuple lookup first. On IPv6 it can also be hit. __udp_disconnect() does not clear sk_v6_daddr, so udp_v6_rehash() files the entry under the peer the socket was connected to with a zero dport, and inet6_match() compares that same field: a datagram from the former peer with a zero source port matches, and source port zero is accepted on receive. On IPv4 the peer is cleared, so a match would need a zero source address as well, which the routing layer rejects as martian. The stale sk_v6_daddr is a separate defect, not addressed here; removing the entry closes this path either way. The entry can also be relocated. __udp_disconnect() clears sk_bound_dev_if, so a subsequent SO_BINDTODEVICE calls ->rehash(), and because the receive address is still specific udp_lib_rehash() moves the entry instead of removing it, into the bucket that (rcv_saddr, num, 0, 0) hashes to -- a pure function of the address and port, so every socket reaching this state on one address and port collects in one bucket. The bucket cannot be chosen from outside, as udp_eshashfn() is seeded with a per-boot secret. This last one became reachable only with commit 644f9108f3a5 ("udp: Make rehash4 independent in udp_lib_rehash()"), which moved the hash4 handling out of a branch a disconnected socket does not take; the stale entry itself dates from the commit in Fixes. Take the socket out of the table before __udp_disconnect() runs, while it still matches how it was filed. This also reaches the wildcard case ahead of udp_lib_rehash()'s udp_unhash4() branch, leaving that branch unreachable from udp_disconnect(); removing it belongs in net-next. udp_disconnect() and udp_abort() are the only UDP entries into __udp_disconnect(), which is shared with raw, ping and l2tp sockets that are not struct udp_sock: ping_prot.obj_size is sizeof(struct inet_sock), so udp_hashed4() on one would read past the allocation.



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/9e95b1a94c9c49b4ba722251bbca2759b9c51737	Additional reference	git.kernel.org/stable/c/c21daf37d5ce9a4e234b7c42238ac1e6041e28e4
Additional reference	git.kernel.org/stable/c/d1b5fd969cc53f29d02b82b05a45135de4a6087a	NVD	nvd.nist.gov/vuln/detail/CVE-2026-98595
ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-103640	Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98595

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