

CVE-2026-98414

Microsoft memcg: avoid charging the root memcg from obj_cgroup_charge_pages()

In the Linux kernel, the following vulnerability has been resolved: memcg: avoid charging the root memcg from obj_cgroup_charge_pages() obj_cgroup_charge_pages() resolves the objcg to its memcg and calls try_charge_memcg(), which does not short circuit the root memcg. That memcg can be the root memcg: obj_cgroup_is_root() reflects the memcg the objcg was created for and is never updated, while memcg_reparent_objcgs() does redirect objcg->memcg to the parent on rmdir.

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 10.1st percentile · FIRST	CISA DEADLINE — No CISA deadline
---	---------------------------------------	--	--

Signals behind the score

☒ counted in the score ☐ not counted

evidence for 2 of 8 signals

☐ CISA KEY	Not currently CISA KEY-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 KEY 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)

An objcg of a dying child of root therefore passes every obj_cgroup_is_root() filter but resolves to the root memcg. Folios keep the objcg they were charged with, so this is easy to reach through zswap: allocate anon memory in a cgroup, move the task out, remove the cgroup, then write to the root cgroup's memory.reclaim. The reclaimed folios are charged through the reparented objcg and end up in refill_stock() with the root memcg: WARNING: mm/memcontrol.c:2198 at refill_stock+0x644/0x940 refill_stock+0x644/0x940 try_charge_memcg+0x12d6/0x1570 __obj_cgroup_charge+0x35/0xf0 obj_cgroup_charge+0x1de/0x210 obj_cgroup_charge_zswap+0x83/0x270 zswap_store+0x1620/0x2000 swap_writeout+0x94c/0x14c0 shrink_folio_list+0x3388/0x52b0 [...] try_to_free_mem_cgroup_pages+0x30d/0x830 user_proactive_reclaim+0x504/0x840 memory_reclaim+0x1f/0x30 Beyond the warning, the charge is asymmetric: obj_cgroup_uncharge_pages() skips refill_stock() for the root memcg, so the root's page counter grows and is never uncharged. It is not user visible, since memory.current is not exposed on the root, but it is a leak. Use try_charge(), which returns early for the root memcg, restoring the symmetry with obj_cgroup_uncharge_pages(). The above sequence was scripted into a standalone reproducer (zswap on, swap on a virtio disk, 512MB of anon memory faulted in inside a child of the root cgroup, the task then migrated to the root cgroup, the child removed, followed by "echo 600M swappiness=max > memory.reclaim" on the root) and run in a CONFIG_DEBUG_VM=y VM. It reproduces the splat on the first zswap store of a reparented folio, with the same call chain as the report. With this patch applied the splat is gone while the zswap store count over the run is unchanged, so the same path is still exercised. cgroup selftests test_zswap, test_kmem and test_memcontrol show no new failures.



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/6e673d0879ef78c395cfe0d3ba316690a60055d8	Additional reference	git.kernel.org/stable/c/d5cdff14c385bf42d03c0dd4c3e88ddf4c28b8a9
Additional reference	git.kernel.org/stable/c/ecc03877b1b05cb1e3a72175ccb868383eccc518	NVD	nvd.nist.gov/vuln/detail/CVE-2026-98414
ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-101063	Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98414

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