

CVE-2026-98818

Microsoft mm/slab: do not wake up kswapd in __kfree_rcu_sheaf()

In the Linux kernel, the following vulnerability has been resolved: mm/slab: do not wake up kswapd in __kfree_rcu_sheaf() Since kfree_rcu() can be called under pi_lock (a raw spinlock in the scheduler), kfree_rcu() itself should never allocate memory with __GFP_KSWAPD_RECLAIM as waking up kswapd ends up acquiring pi_lock, which leads to a deadlock. Reproducing the issue even intentionally was not straightforward.

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 5.1st 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)

The set_cpus_allowed_force() path that is called under pi_lock is exercised very rarely, and competing tasks that allocate memory will most likely wake kswapd up. Therefore the existence of the deadlock was verified with a modified kernel that has a lockdep map for waking up kswapd: ===== WARNING: possible circular locking dependency detected 7.2.0-rc1-slab-for-next+ #10 Not tainted

```
----- git/6660 is trying to acquire lock: ffff8e524533c550 (&p->pi_lock){-.-}
{2:2}, at: _raw_spin_lock_irqsave+0x12/0x20 but task is already holding lock: ffff8e57bffff1a0 (&pgdat->kswapd_wait){....}-
{3:3}, at: _raw_spin_lock_irqsave+0x12/0x20 which lock already depends on the new lock. the existing dependency chain (in
reverse order) is: -> #2 (&pgdat->kswapd_wait){....}- {3:3}: __lock_acquire+0x5a4/0xc50 lock_acquire.part.0+0xb7/0x240
lock_acquire+0x70/0x170 __raw_spin_lock_irqsave+0x44/0x80 _raw_spin_lock_irqsave+0x12/0x20
__wake_up_common_lock+0x31/0xa0 __wake_up+0x20/0x40 kswapd_wakeup_wake+0x7e/0x110
wakeup_kswapd+0x295/0x320 wake_all_kswapds+0xac/0x1a0 __alloc_pages_slowpath.constprop.0+0x282/0xf80
__alloc_frozen_pages_noprof+0x32a/0x360 alloc_slab_page+0x2e/0x160 allocate_slab+0x82/0x420 new_slab+0x52/0xb0
refill_objects+0x13d/0x190 refill_sheaf+0x5d/0xd0 __pcs_replace_empty_main+0x230/0xb10 [...] -> #1 (kswapd_wakeup)
{-.-}-{0:0}: __lock_acquire+0x5a4/0xc50 lock_sync.part.0+0x75/0x100 lock_sync+0x36/0x70
might_wakeup_kswapd+0x61/0xa0 __kfree_rcu_sheaf+0x33/0xd90 kvfree_call_rcu+0x1d4/0x3b0
set_cpus_allowed_force+0x163/0x220 cpuset_cpus_allowed_fallback+0x18b/0x240 select_fallback_rq+0x1e6/0x250 [...] -> #0
(&p->pi_lock){-.-}-{2:2}: check_prev_add+0xe6/0xe00 validate_chain+0x51e/0x6e0 __lock_acquire+0x5a4/0xc50
lock_acquire.part.0+0xb7/0x240 lock_acquire+0x70/0x170 __raw_spin_lock_irqsave+0x44/0x80
_raw_spin_lock_irqsave+0x12/0x20 try_to_wake_up+0x77/0xa90 default_wake_function+0x27/0x60
autoremove_wake_function+0x23/0xb0 __wake_up_common+0xb8/0x170 __wake_up_common_lock+0x51/0xa0
__wake_up+0x20/0x40 kswapd_wakeup_wake+0x7e/0x110 wakeup_kswapd+0x295/0x320 wake_all_kswapds+0xac/0x1a0
__alloc_pages_slowpath.constprop.0+0x282/0xf80 __alloc_frozen_pages_noprof+0x32a/0x360 alloc_slab_page+0x2e/0x160
allocate_slab+0x82/0x420 new_slab+0x52/0xb0 refill_objects+0x13d/0x190 refill_sheaf+0x5d/0xd0
```

__pcs_replace_empty_main+0x230/0xb10 [...] other info that might help us debug this: Chain exists of: &p->pi_lock --> kswapd_wakeup --> &pgdat->kswapd_wait Possible unsafe locking scenario: CPU0 CPU1 ----- lock(&pgdat->kswapd_wait); lock(kswapd_wakeup); lock(&pgdat->kswapd_wait); lock(&p->pi_lock); *** DEADLOCK *** 3 locks held by git/6660: #0: ffff8e519f0e9e20 (&type->i_mutex_dir_key#6){++++}-{4:4}, at: lookup_slow+0x2d/0x60 #1: ffffffff8a31c460 (kswapd_wakeup){-.-}- ---truncated---

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/46545c86b54fffb969b6e3662942b4a6191619eaa	Additional reference	git.kernel.org/stable/c/af2fb3ee6679b517e2fdbd3ba16ec276a8b74312
NVD	nvd.nist.gov/vuln/detail/CVE-2026-98818	ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-103903
Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98818		

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