

CVE-2026-98431

Microsoft RDMA/uverbs: Fix mmap_lock/disassociation_lock circular dependency

In the Linux kernel, the following vulnerability has been resolved: RDMA/uverbs: Fix mmap_lock/disassociation_lock circular dependency Commit 51976c6cd786 ("RDMA/core: Provide rdma_user_mmap_disassociate() to disassociate mmap pages") introduced disassociation_lock to protect new mmap registrations against uverbs_user_mmap_disassociate(), but created an ABBA deadlock: Thread A (mmap / fork): mmap_lock -> disassociation_lock Thread B (disassociate): disassociation_lock -> mmap_lock Fix by removing

EVRS 0 /100 Informational priority	CVSS — Not scored	EPSS 0.002 9.1st percentile · FIRST	CISA DEADLINE — No CISA deadline
---	---------------------------------------	---	--

Signals behind the score

☒ counted in the score ☐ not counted

evidence for 2 of 8 signals

☐ CISA 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)

disassociation_lock entirely and using the pre-existing hw_destroy_rwlock instead. hw_destroy_rwlock already provides the same protection: rdma_umap_open() and ib_uverbs_mmap() both use down_read_trylock() before registering a new VMA, so holding hw_destroy_rwlock in uverbs_user_mmap_disassociate() is sufficient to block new registrations. trylock is used in both mmap paths (not blocking down_read) because mmap_lock is already held on entry, and uverbs_user_mmap_disassociate() acquires mmap_lock internally — a blocking read would recreate the same deadlock. The only caller that was not taking hw_destroy_rwlock for write was rdma_user_mmap_disassociate(). Fix it to take the rwsem per-ufile while iterating under lists_mutex. This is safe because ib_uverbs_close() releases hw_destroy_rwlock entirely before acquiring lists_mutex, so the two locks are never held simultaneously. lockdep warning: [776.654252]

```
===== [ 776.655214] WARNING: possible circular locking
dependency detected [ 776.656167] 6.18.0for-upstream_debug_94e244d9ccab #1 Not tainted [ 776.657114]
----- [ 776.658087] devlink/14824 is trying to acquire lock: [ 776.658879]
ffff88811170c800 (&mm->mmap_lock){++++}-{4:4}, at: uverbs_user_mmap_disassociate+0x168/0x780 [ib_uverbs]
[ 776.660479] [ 776.660479] but task is already holding lock: [ 776.661460] ffff888142d92b08 (&file->disassociation_lock)
{+.-}{4:4}, at: uverbs_user_mmap_disassociate+0x39/0x780 [ib_uverbs] [ 776.663177] [ 776.663177] which lock already
depends on the new lock. [ 776.663177] [ 776.664525] [ 776.664525] the existing dependency chain (in reverse order) is:
[ 776.665724] [ 776.665724] -> #2 (&file->disassociation_lock){+.-}{4:4}: [ 776.666887] __mutex_lock+0x16d/0x2330
[ 776.667633] rdma_umap_open+0x129/0x280 [ib_uverbs] [ 776.668489] dup_mmap+0xa40/0x1790 [ 776.669170]
copy_process+0x5dd2/0x6170 [ 776.669933] kernel_clone+0xb6/0x610 [ 776.670636] __do_sys_clone+0xb5/0xf0
[ 776.671354] do_syscall_64+0x70/0x12e0 [ 776.672083] entry_SYSCALL_64_after_hwframe+0x4b/0x53 [ 776.672940]
[ 776.672940] -> #1 (&mm->mmap_lock/1){+.-}{4:4}: [ 776.673985] down_write_nested+0x90/0x1e0 [ 776.674751]
dup_mmap+0x201/0x1790 [ 776.675448] copy_process+0x5dd2/0x6170 [ 776.676180] kernel_clone+0xb6/0x610
[ 776.676904] __do_sys_clone+0xb5/0xf0 [ 776.677615] do_syscall_64+0x70/0x12e0 [ 776.678351]
```

```
entry_SYSCALL_64_after_hwframe+0x4b/0x53 [ 776.679239] [ 776.679239] -> #0 (&mm->mmap_lock){++++}--{4:4}:
[ 776.680253] __lock_acquire+0x18c6/0x2ec0 [ 776.681018] lock_acquire+0x10e/0x2e0 [ 776.681742]
down_read+0x95/0x430 [ 776.682395] uverbs_user_mmap_disassociate+0x168/0x780 [ib_uverbs] [ 776.683436]
uverbs_destroy_ufile_hw+0x1ae/0x270 [ib_uverbs] [ 776.684416] ib_uverbs_remove_one+0x22b/0x420 [ib_uverbs]
[ 776.685371] remove_client_context+0xa6/0xf0 [ib_core] [ 776.686342] disable_device+0x12b/0x240 [ib_core]
[ 776.687249] __ib_unregister_device+0x269/0x460 [ib_core] [ 776.688233] ib_unregister_device+0x21/0x30 [ib_core]
[ 776.689140] mlx5r_remove+0xd0/0x170 [mlx5_ib] [ 776.689999] device_release_driver_internal+0x3b2/0x5 ---
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/426f54f20b79c5b164a0472ad3f6ba36afcd08c6	Additional reference	git.kernel.org/stable/c/a44a3f175eae7e5aeb6a8fed381c4a0d5f49236
NVD	nvd.nist.gov/vuln/detail/CVE-2026-98431	ENISA EUVD	euvd.enisa.europa.eu/vulnerability/EUVD-2026-101024
Current data	evulnable.com/vulnerability-management/cve/CVE-2026-98431		

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