

# CVE-2026-98256

## Microsoft signal: Prevent exec() race

In the Linux kernel, the following vulnerability has been resolved: signal: Prevent exec() race Hyunwoo debugged the following KASAN UAF splat: BUG: KASAN: slab-use-after-free in \_\_send\_signal\_locked+0xb27/0xba0 Write of size 8 at addr ffff888007ed80c8 by task poc/79 ... Call Trace: \_\_send\_signal\_locked+0xb27/0xba0 do\_send\_sig\_info+0xa7/0x160 do\_send\_specific+0x76/0xa0 \_\_x64\_sys\_tgkill+0x193/0x270 ... Allocated by task 80: do\_timer\_create+0x1a4/0x1030 \_\_x64\_sys\_timer\_create+0x145/0x190 ...

|                                                         |                                       |                                                  |                                               |
|---------------------------------------------------------|---------------------------------------|--------------------------------------------------|-----------------------------------------------|
| EVRS<br><b>12</b> / <sub>100</sub><br>Elevated priority | CVSS V3.1<br><b>7.8</b><br>High · NVD | EPSS<br><b>0.001</b><br>2.5th percentile · FIRST | CISA DEADLINE<br><b>—</b><br>No CISA deadline |
|---------------------------------------------------------|---------------------------------------|--------------------------------------------------|-----------------------------------------------|

## Signals behind the score

☒ counted in the score ☐ not counted

evidence for 3 of 8 signals

|                                                 |                                    |                                                                |                                                                                                    |
|-------------------------------------------------|------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> <b>CISA KEV</b>        | Not currently CISA KEV-listed      | <input type="checkbox"/> <b>Vendor exploitation</b>            | Active-exploitation status not yet checked - scored at the population base rate (none) until known |
| <input type="checkbox"/> <b>Ransomware</b>      | Ransomware association unavailable | <input type="checkbox"/> <b>KEV recency</b>                    | CISA KEV listing date unavailable                                                                  |
| <input type="checkbox"/> <b>EPSS</b>            | 0.001 probability                  | <input type="checkbox"/> <b>Pre-patch disclosure</b>           | Disclosure status unavailable                                                                      |
| <input checked="" type="checkbox"/> <b>CVSS</b> | Base score 7.8                     | <input type="checkbox"/> <b>Microsoft Exploitability Index</b> | Not a Microsoft CVE                                                                                |

## Remediation

**Scheduled Maintenance** Patch or fixed version identified

Basis: External remediation intelligence

Freed by task 12: kmem\_cache\_free\_bulk+0x1f8/0x4a0 kvfree\_rcu\_bulk+0x14f/0x1c0 kfree\_rcu\_work+0x128/0x1a0 ... Last potentially related work creation: kvfree\_call\_rcu+0x39/0x390 \_\_flush\_itimer\_signals+0x211/0x320 flush\_itimer\_signals+0x47/0x90 begin\_new\_exec+0xa6b/0x28c0 It turned out that this happens with a non-leader exec() as Hyunwoo explained: de\_thread() calls exchange\_tids() before release\_task(leader), so the struct pid held by a SIGEV\_THREAD\_ID timer created against the leader's tid now points to the thread which called execve(). pid\_task() returns that thread and lock\_task\_sighand() on it succeeds. If the timer signal is blocked, its sigqueue stays queued on the leader's task::pending. The next expiry of that timer can then run while release\_task() flushes the queue. posixtimer\_send\_sigqueue() checks whether the sigqueue is already queued with a plain list\_empty(), which only reads list\_head::next. list\_del\_init() is not atomic and INIT\_LIST\_HEAD() stores list\_head::next before list\_head::prev, so the check can pass in between. list\_add\_tail() queues the entry on the task::pending of the live thread, and the list\_head::prev store from the flush then overwrites the list\_head::prev link that list\_add\_tail() has just set. \_\_flush\_itimer\_signals() does not undo that either. With list\_head::prev pointing at the entry itself, its list\_del\_init() only stores the same values again, so the entry is not removed from the list. It is still there after the last reference is dropped and the timer is freed by RCU, and the list\_add\_tail() of a later tgkill() follows that list\_head::prev into the freed timer. This problem surfaced with the recent commit which moved the sigqueue flush out of the sighand lock held region. Hyunwoo proposed to fix this by using list\_del\_init\_careful(), but that just papers over the problem. After some discussions and various attempts to solve it, Eric pointed out that there is no reason to flush task::pending late in release\_task() and it should be done in exit\_signals() already. As nothing can collect and deliver signals which are queued in a dying task's pending queue, there is no reason to delay it further. But it has to be ensured that no signals can be queued into it after that point. exit\_signals() sets PF\_EXITING in task::flags, which can be used as an indicator for this. Cure it by: - Preventing signal queueing for task private signals (PIDTYPE\_PID) when the task has PF\_EXITING set in \_\_send\_signal\_locked() and in posixtimer\_send\_sigqueue(). - Protecting the unlocked setting of PF\_EXITING in exit\_signals() for the task group empty and the group exit case with sighand lock - Flushing task::pending signals right there. Optimize that by moving the whole pending list to an on-stack list head under sighand lock and free the signals without the lock held. There has been quite some discussion about the lockless flush and the non-leader exec case on weakly ordered systems. The problem is that a third party

which tries to send a posix timer signal relies on the PID lookup to find the target task and that lookup might result in the new leader when the signal was originally directed to the old leader. In case that the signal was queued on the old leader then the lockless flush raised a concern over the following situation: old\_leader new\_leader third party A: flush\_list() // list\_del\_in --- truncated---

## Affected and fixed versions

| PACKAGE                                 | AFFECTED | FIXED IN                | SOURCE |
|-----------------------------------------|----------|-------------------------|--------|
| Not identified by the connected sources | –        | no fixed version listed | NVD    |

## Sources

|                            |                                                                                                                                                         |                            |                                                                                                                                                         |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Vendor guidance</b>     | <a href="https://git.kernel.org/stable/c/621c365739828900f32a0c1cdfe5387528a62176">git.kernel.org/stable/c/621c365739828900f32a0c1cdfe5387528a62176</a> | <b>Additional guidance</b> | <a href="https://git.kernel.org/stable/c/934bd95f1a6404dbb845951b823547abab05a649">git.kernel.org/stable/c/934bd95f1a6404dbb845951b823547abab05a649</a> |
| <b>Additional guidance</b> | <a href="https://git.kernel.org/stable/c/d2710c8d938ae6a825a6463158e6e6f31eac792a">git.kernel.org/stable/c/d2710c8d938ae6a825a6463158e6e6f31eac792a</a> | <b>NVD</b>                 | <a href="https://nvd.nist.gov/vuln/detail/CVE-2026-98256">nvd.nist.gov/vuln/detail/CVE-2026-98256</a>                                                   |
| <b>ENISA EUVD</b>          | <a href="https://euvd.enisa.europa.eu/vulnerability/EUVD-2026-93184">euvd.enisa.europa.eu/vulnerability/EUVD-2026-93184</a>                             | <b>Current data</b>        | <a href="https://evulnable.com/vulnerability-management/cve/CVE-2026-98256">evulnable.com/vulnerability-management/cve/CVE-2026-98256</a>               |

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