PatchMon PATCHMON BLOG

Copy Fail (CVE-2026-31431): What It Is, Why It Matters, and How to Patch Your Linux Fleet

A practical breakdown of the Copy Fail Linux kernel privilege escalation flaw. How it works, which kernels are affected, what the distros are shipping, and how to confirm your servers are actually patched.

AUTHOR

PatchMon Team

PUBLISHED

2 May 2026

CATEGORIES

Security, Patch Management

READ ONLINE

https://patchmon.net/blog/copy-fail-cve-2026-31431-linux-kernel-patch

Contents

Here is a Wednesday afternoon scenario for you.

Someone on your team forwards a Slack message with a CVE number, a CVSS score starting with "7", and a Wikipedia entry. The CVE has a catchy nickname. The Wikipedia entry has been live for less than a week. The message ends with: *"are we exposed?"*

This time the CVE is CVE-2026-31431 (https://nvd.nist.gov/vuln/detail/CVE-2026-31431), nicknamed **Copy Fail**, and for most Linux fleets the honest answer is yes. If you run anything other than a freshly cut Ubuntu 26.04 box, you almost certainly have at least one server that an unprivileged local user can take to root with a 732-byte Python script.

This post is the calm, practical version of that conversation. What the bug actually is, why it is more concerning than the usual kernel CVE, which kernels are affected, what your distro is shipping, and how to verify your fleet is patched rather than just hoping it is.

The Short Version

If you only have two minutes:

> **What it is:** A local privilege escalation in the Linux kernel's `algif_aead` module, the userspace crypto interface exposed via `AF_ALG` sockets.
>
> **Severity:** CVSS 7.8 High, local, low complexity, no user interaction (https://nvd.nist.gov/vuln/detail/CVE-2026-31431).
>
> **Disclosed:** 29 April 2026.
>
> **Who is affected:** Practically every mainline Linux kernel built between 2017 and the patch landing in early April 2026. That is most distros currently in production, including Ubuntu, Debian, RHEL, Rocky, AlmaLinux, Amazon Linux, SUSE, Fedora, and Arch.
>
> **What an attacker gets:** Root, locally, in seconds, with no race condition and no version-specific tuning.
>
> **What you do about it:** Update your kernel from your distro's security feed and reboot. If you cannot reboot today, blacklist the `algif_aead` module as a temporary mitigation.

The rest of this post is the why and the how.

What Copy Fail Actually Is

The Linux kernel exposes its cryptographic primitives to userspace through a socket family called `AF_ALG`. It is the interface that lets non-root processes ask the kernel to do things like AEAD encryption, often to take advantage of hardware acceleration. The module that handles authenticated-encryption-with-associated-data requests is called `algif_aead`.

Back in 2017, somebody made a perfectly reasonable performance change. Rather than copy buffers around, the kernel started doing AEAD work in place, with the destination scatterlist pointing at the same pages as the source. Faster, less memory, fewer copies. Sensible.

The problem is what happens when one of those source pages is a page-cache page. The page cache is the kernel's in-memory copy of files on disk, and it is shared between processes. When you `splice()` data out of a file into the crypto pipeline, the kernel uses the actual cache page rather than a copy. With the in-place optimisation in effect, that cache page also becomes the destination. The kernel happily writes the AEAD tag into it, bypassing every file permission check that would normally apply.

In other words: an unprivileged user can convince the kernel to perform a controlled write into the in-memory copy of a file they only have read access to. Including, for example, `/usr/bin/su`.

Sysdig's technical writeup (https://www.sysdig.com/blog/cve-2026-31431-copy-fail-linux-kernel-flaw-lets-local-users-gain-root-in-seconds) walks through the exact four-step exploitation path: bind an `AF_ALG` socket to `authencesn(hmac(sha256),cbc(aes))`, splice setuid binary pages into the crypto context, send a `recvmsg()` with carefully chosen associated-data bytes, then execute the now-corrupted setuid binary. The published proof of concept does this in about ten lines of Python and 732 bytes of source. The researchers at Xint published the writeup (https://xint.io/blog/copy-fail-linux-distributions) that gave the bug its nickname, and The Register covered (https://www.theregister.com/2026/04/30/linux_cryptographic_code_flaw/) the public disclosure.

Why This One Hurts More Than the Usual Kernel CVE

Kernel privilege escalations are not new. We have lived through Dirty COW, Dirty Pipe, and dozens of OverlayFS bugs. So why does this one merit a dedicated blog post?

Three reasons.

It is deterministic. Dirty Pipe (CVE-2022-0847) (https://www.cisa.gov/news-events/alerts/2022/03/10/dirty-pipe-privilege-escalation-vulnerability-linux) needed precise pipe buffer manipulation and version-specific offsets. Copy Fail does not. There is no race window, no spray, no retry loop. The same script runs successfully on Ubuntu 24.04, Amazon Linux 2023, RHEL, SUSE, and anything else you point it at, regardless of CPU architecture. If your kernel is in the affected range and `algif_aead` is loadable, the exploit works on the first try.

It is small. A 732-byte Python script with no exotic dependencies fits inside almost any constrained environment. CI/CD job containers, Kubernetes pods, jump boxes, build agents, anywhere a user can execute Python. Bugcrowd's summary (https://www.bugcrowd.com/blog/what-we-know-about-copy-fail-cve-2026-31431/) calls out this footprint specifically because of how easy it is to drop into a compromised low-privilege foothold and turn it into root.

The blast radius is broad. CERT-EU's advisory (https://cert.europa.eu/publications/security-advisories/2026-005/) recommends prioritising Kubernetes nodes and CI/CD runners. That is not a coincidence. Any environment where you let untrusted or semi-trusted code run as a non-root user, on a kernel that predates the fix, is now an environment where that code can probably reach root.

This is not a "patch it sometime this quarter" CVE. It is a "patch it before someone with a working exploit gets a foothold" CVE.

Which Kernels Are Affected

The introducing change landed in 2017, so any kernel from roughly 4.14 onwards is in scope until the fix is applied. Per Sysdig's technical analysis (https://www.sysdig.com/blog/cve-2026-31431-copy-fail-linux-kernel-flaw-lets-local-users-gain-root-in-seconds), the relevant ranges are:

> **Vulnerable:** 4.14 through to 7.0-rc, plus any unpatched 6.18.x, 6.19.x, 6.12.x, 6.6.x, 5.15.x, and 5.10.x LTS kernels.
>
> **Fixed mainline:** 7.0, 6.19.12, and 6.18.22.

Backports to older LTS branches are landing on a rolling basis, which is why your distro's package version matters more than the kernel version string itself.

What Your Distro Is Shipping

This is the part that changes daily, so check vendor advisories before you act. As of disclosure week:

Ubuntu. The Ubuntu Security Team's advisory (https://ubuntu.com/blog/copy-fail-vulnerability-fixes-available) confirms 26.04 ships an unaffected kernel, and earlier releases are receiving a `kmod` package update that disables the vulnerable `algif_aead` module while the kernel patch makes its way through the staging process. If you are on Ubuntu 16.04 LTS or later with `unattended-upgrades` enabled, the mitigation lands automatically within 24 hours. The mitigation does carry a small regression risk for any application that genuinely uses kernel-side AEAD acceleration and cannot fall back to userspace crypto.

Red Hat / RHEL / Rocky / Alma. AlmaLinux published its patched kernels on 1 May 2026 (https://almalinux.org/blog/2026-05-01-cve-2026-31431-copy-fail/), built directly against the upstream fix series while waiting for Red Hat's official errata. CloudLinux published its own kernel update guidance (https://blog.cloudlinux.com/cve-2026-31431-copy-fail-kernel-update). Watch RHSA channels for Red Hat's official advisory.

SUSE, Debian, Fedora, Amazon Linux, Arch. All confirmed affected, all shipping fixes through normal security update channels. If your distro has a security mailing list, this is the week to read it.

Microsoft. Microsoft published guidance (https://www.microsoft.com/en-us/security/blog/2026/05/01/cve-2026-31431-copy-fail-vulnerability-enables-linux-root-privilege-escalation/) for Azure customers running Linux workloads, and the Hacker News writeup (https://thehackernews.com/2026/04/new-linux-copy-fail-vulnerability.html) covers the cloud-side implications more broadly.

The takeaway: your distro maintainers are ahead of you. The hard part is not finding the patch. The hard part is making sure it actually got applied to every server in your fleet, and that every one of those servers has been rebooted onto the new kernel.

If You Cannot Patch Right Now

Sometimes you genuinely cannot reboot a kernel today. Maintenance windows exist for a reason, and a forced kernel reboot during business hours is a perfectly good way to turn one CVE into a P1 incident.

For those cases, both CERT-EU (https://cert.europa.eu/publications/security-advisories/2026-005/) and the distro security teams agree on the same short-term mitigation: stop the `algif_aead` module from loading.

```
# Blacklist the vulnerable module
echo "install algif_aead /bin/false" | sudo tee /etc/modprobe.d/disable-algif-aead.conf

# If it is currently loaded, unload it
sudo modprobe -r algif_aead 2>/dev/null || true

# Confirm
lsmod | grep algif_aead
```

This will break any application that explicitly depends on `AF_ALG` AEAD operations, but the vast majority of Linux software falls back to userspace crypto cleanly. Test in a staging environment first if you can.

For containerised environments, the more robust mitigation is to deny `AF_ALG` socket creation at the seccomp profile level. The default `docker` and `containerd` profiles already restrict `AF_ALG` in many configurations; verify yours does, especially on Kubernetes nodes that schedule untrusted workloads.

None of this is a substitute for patching. It is a substitute for patching *right now*, while you organise a maintenance window for the proper fix.

How to Tell Your Fleet Is Actually Patched

The uncomfortable truth about kernel CVEs is that "we ran apt upgrade last night" is rarely the same as "every server is on a fixed kernel." Three things tend to go wrong:

The package was installed, but the server was not rebooted. The new kernel sits on disk while the running kernel is still vulnerable. `uname -r` says one thing, the package manager says another.

A handful of servers were missed. The forgotten staging box, the long-running build agent, the bastion that someone manually pinned a kernel on six months ago.

The advisory landed at midnight and your unattended-upgrades window is at 03:00 the next day. Some boxes patched themselves, some did not, and nobody is currently checking which is which.

This is exactly the "did we actually patch it?" problem PatchMon (https://patchmon.net) was built for. PatchMon's agents report the running kernel, the installed kernel package, and the available updates from every server in your fleet to a single dashboard, so a query like *"show me every host whose running kernel does not match the installed kernel and which still has* `algif_aead` *loadable"* takes seconds to answer rather than an afternoon of SSH loops. When a CVE like Copy Fail drops, the workflow is: filter by affected distro, filter by kernel version range, hit reboot or schedule a window, watch the count fall to zero. There is also a complete history with timestamps, which is what your auditor will ask for the next time the patch evidence question comes up.

The fastest way to evaluate is a PatchMon Cloud trial (https://patchmon.net/pricing). It runs on real hosts, billing tracks the hosts that actually check in rather than a fleet tier, and the trial is 14 days with no charge if you cancel before day 14. If you would rather self-host, the Community Edition (https://patchmon.net/open-source) is AGPLv3 and free; you can have it running on your own infrastructure in under half an hour.

Bottom Line

Copy Fail is a serious, deterministic, broadly applicable Linux privilege escalation. The good news is that the upstream fix exists, your distro has either shipped a patched kernel or is about to, and the temporary mitigation is a one-line modprobe rule.

The thing to focus on this week:

Apply your distro's kernel update on every host that runs Linux. Not most. Every.

Reboot. A patched kernel package on disk does nothing for you until the box is running on it.

Verify. Compare running kernel against installed kernel, across the entire fleet, on a single timestamped report. Not by SSH-ing in to spot-check.

For environments where untrusted code can run as a non-root user, like Kubernetes nodes scheduling untrusted workloads, CI/CD runners executing third-party builds, and shared developer boxes, treat this as a P1 until those boxes are confirmed on a fixed kernel.

CVEs like Copy Fail are not rare. The teams that handle them calmly are not the ones with the most expensive tooling. They are the ones who can answer *"is every server patched, with evidence?"* on the same day the question is asked. Whatever tool you use to do that, this is a good week to make sure it actually works.

If you want to see how PatchMon handles a CVE response across a Linux fleet, the 14-day Cloud trial (https://patchmon.net/pricing) is the quickest way in, and the Community Edition (https://patchmon.net/open-source) is free to deploy on your own stack. Either way, get your kernels patched first. The rest is housekeeping.

Sources and Further Reading

- NVD: CVE-2026-31431 detail (https://nvd.nist.gov/vuln/detail/CVE-2026-31431)
- CERT-EU advisory: High Vulnerability in the Linux Kernel ("Copy Fail") (https://cert.europa.eu/publications/security-advisories/2026-005/)
- Sysdig: CVE-2026-31431 "Copy Fail" Linux kernel flaw lets local users gain root in seconds (https://www.sysdig.com/blog/cve-2026-31431-copy-fail-linux-kernel-flaw-lets-local-users-gain-root-in-seconds)
- Xint: Copy Fail, 732 Bytes to Root on Every Major Linux Distribution (https://xint.io/blog/copy-fail-linux-distributions)
- Ubuntu: Fixes available for CVE-2026-31431 (Copy Fail) (https://ubuntu.com/blog/copy-fail-vulnerability-fixes-available)
- AlmaLinux: Copy Fail (CVE-2026-31431) patches released (https://almalinux.org/blog/2026-05-01-cve-2026-31431-copy-fail/)
- CloudLinux: CVE-2026-31431 (Copy Fail) kernel update (https://blog.cloudlinux.com/cve-2026-31431-copy-fail-kernel-update)
- Microsoft Security Blog: CVE-2026-31431 Copy Fail vulnerability enables Linux root privilege escalation across cloud environments (https://www.microsoft.com/en-us/security/blog/2026/05/01/cve-2026-31431-copy-fail-vulnerability-enables-linux-root-privilege-escalation/)
- The Hacker News: New Linux 'Copy Fail' Vulnerability Enables Root Access (https://thehackernews.com/2026/04/new-linux-copy-fail-vulnerability.html)
- The Register: Linux cryptographic code flaw offers fast route to root (https://www.theregister.com/2026/04/30/linux_cryptographic_code_flaw/)
- Bugcrowd: What we know about Copy Fail (CVE-2026-31431) (https://www.bugcrowd.com/blog/what-we-know-about-copy-fail-cve-2026-31431/)

PatchMon

ABOUT PATCHMON

The open source Linux patch management platform

PatchMon gives sysadmins one dashboard for patching across Linux, FreeBSD, and Windows fleets. Run it as a managed SaaS on PatchMon Cloud (per-host billing, 14-day trial, no fleet minimum) or self-host the AGPLv3 Community Edition on your own infrastructure.

Start a trial: patchmon.net/pricing