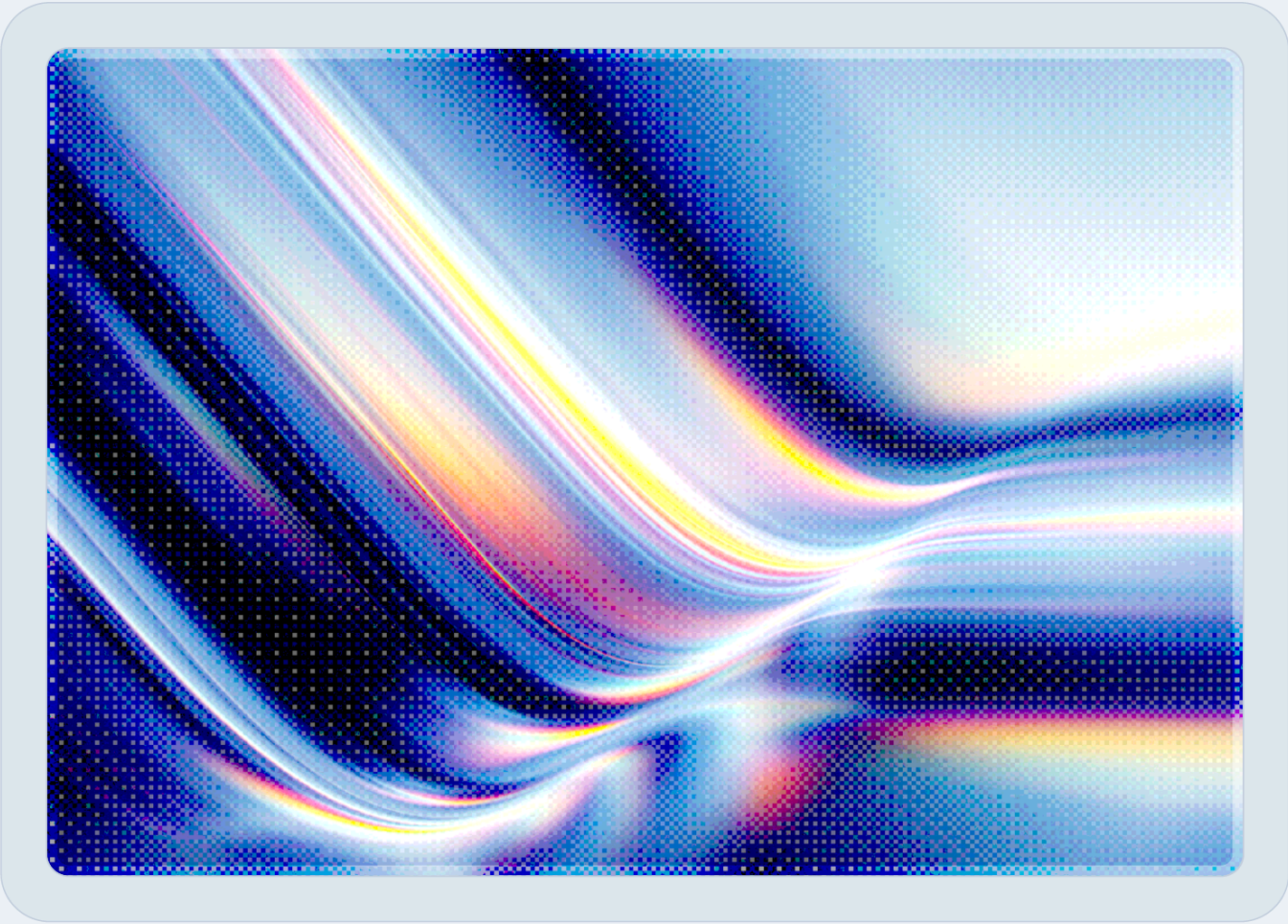
CSAI Foundation | Cloud Security Alliance

Copy Fail (CVE-2026-31431): Linux Root Escalation Under Active Exploitation

2026-05-03

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- CVE-2026-31431, disclosed April 29, 2026, is a local privilege escalation vulnerability in the Linux kernel's cryptographic subsystem affecting all tested major distributions running kernels released since 2017 [1][4].
- A publicly released 732-byte Python proof-of-concept exploit grants any unprivileged local user full root access without requiring a race condition, timing precision, or user interaction, and has been confirmed to work reliably across Ubuntu, Amazon Linux, RHEL, and SUSE [9] [10].
- The vulnerability enables container escape in Kubernetes environments because the Linux kernel's page cache — the mechanism the attack exploits — is shared across all processes and containers on the same host, allowing a compromised pod to escalate to node root [6][12].
- Vendor patches are in active distribution; organizations should apply kernel updates immediately and prioritize Kubernetes nodes, CI/CD runners, and any Linux hosts running multi-tenant or AI workloads [3][5][14].
- On RHEL-family distributions where the vulnerable module is compiled into the kernel and cannot be unloaded, blocking `AF_ALG` socket creation via seccomp policy is the most broadly applicable interim protection [2][7].

Background

The Linux kernel includes a userspace cryptographic API known as `AF_ALG`, which exposes kernel-side cryptographic operations — ciphers, message authentication codes, and authenticated encryption schemes — to unprivileged applications through a socket interface. This API is intended to allow software to leverage hardware cryptographic acceleration without requiring applications to run kernel-mode code. It is present and enabled by default across virtually all mainstream Linux distributions. Over the course of nine years, three incremental kernel changes converged to introduce a critical logic error in one component of this interface.

The first of those changes arrived in 2011 with the addition of `authencesn`, an Authenticated Encryption with Associated Data (AEAD) wrapper used primarily by IPsec to handle extended sequence numbers. The second arrived in 2015, when the kernel added AEAD socket support to the `AF_ALG` interface, making `authencesn` operations accessible from userspace through a new module, `algif_aead`. The third and final ingredient appeared in 2017: a performance optimization that modified `algif_aead` to perform AEAD operations in-place rather than using a separate output buffer [4][10]. Taken in isolation, each of these changes appeared correct. In combination, the 2017 optimization introduced a logic flaw: when the `authencesn` template is used through the `algif_aead` in-place path, the kernel can be made to write four controlled bytes to an attacker-chosen location in the page cache of any readable file on the system.

The vulnerability was publicly disclosed on April 29, 2026, by Theori researcher Taeyang Lee, with the Xint Code Research Team credited for surfacing the flaw through AI-assisted analysis of the kernel's `crypto/` subsystem [4][9][11]. Their tooling reportedly identified the bug in approximately one hour of directed scanning — a finding that the kernel community had not surfaced despite years of review of the same codebase, though the flaw is of a type that emerges only from the interaction of changes across multiple years and subsystems rather than from any single commit in isolation. The upstream kernel fix was committed on April 1, 2026, following a coordinated disclosure period, reverting `algif_aead` to out-of-place operation: effectively undoing the 2017 performance optimization while preserving associated data handling improvements that were bundled with it [8]. Vendor distributions including Ubuntu, Red Hat Enterprise Linux, SUSE, and Amazon Linux subsequently issued patched kernel updates, with availability varying by distribution release [3][5][14].

Security Analysis

Technical Mechanism

The controlled 4-byte write that Copy Fail enables targets the Linux kernel's page cache — the in-memory representation of file contents that the kernel maintains on behalf of all processes on the system. Crucially, the attack modifies only the page cache representation of a file, not the file itself on disk. This means the corruption persists only for the duration of the current boot and is visible to every subsequent process that reads or executes the affected file, including processes in other containers or on the host, until the system is rebooted or the page cache entry is evicted [6][9].

The exploit uses this 4-byte write capability to corrupt the in-memory image of a setuid binary — a program such as `sudo` or `passwd` that the kernel executes with elevated privileges regardless of which user invokes it. By modifying specific bytes of the binary's page cache entry, an unprivileged attacker redirects execution into attacker-controlled code when the binary is subsequently invoked, yielding a root shell. The full sequence executes in seconds. A publicly released proof-of-concept accomplishes the entire chain in a 732-byte Python script that requires no kernel debugging environment and no distribution-specific offset tuning [9][10][15]. By contrast with many prior kernel local privilege escalation vulnerabilities — which commonly require kernel version-specific gadgets or depend on race conditions — Copy Fail requires neither, a distinction noted by multiple security researchers [9][10].

The in-memory-only nature of the attack also has significant implications for detection. Traditional file integrity monitoring tools — which compare the on-disk hashes of critical binaries against known-good baselines — will report no anomaly, because the on-disk file is unchanged. There are no modified file timestamps, no audit log entries for file writes, and no persistent indicators of compromise after a reboot. The most reliable detection approach is monitoring at the syscall level for `AF_ALG` socket creation events in processes or containers where such calls would not be expected [6].

Cloud and Container Exposure

The threat profile escalates substantially in cloud and Kubernetes environments because the kernel's page cache is a resource shared by all processes on a host, regardless of container boundaries. Standard container isolation mechanisms — Linux namespaces, cgroups, and the Kubernetes Pod Security Standards profile hierarchy — operate above the kernel level and do not restrict access to `AF_ALG` sockets by default [13]. Independent testing has confirmed that the Kubernetes `RuntimeDefault` seccomp profile, the baseline hardening mechanism recommended for most workloads, does not block the `AF_ALG` socket creation calls on which the exploit depends [13].

An attacker who has achieved code execution within a single container — through a compromised dependency, a malicious CI pipeline job, or an application vulnerability — can use Copy Fail to escalate to root on the Kubernetes node. From node root, the attacker typically gains access to the node's kubelet credentials, the secrets and service account tokens mounted by all pods scheduled to that node, and potentially the Kubernetes control plane depending on the cluster's RBAC configuration [1][6]. In environments where multiple tenants or workloads share a node pool — including AI inference services, training pipelines, and model-serving infrastructure — this represents a complete compromise of the compute substrate and all data accessible to those workloads, including model weights, training datasets, API credentials, and user data being processed at inference time.

The persistence characteristics of the attack compound this risk in containerized environments. Because the page cache modification is in-memory, a containerized attacker who achieves node root can exfiltrate secrets and establish persistence through other means — deploying a DaemonSet, modifying node configuration, or accessing the etcd store — while leaving no trace in the original container's filesystem.

Exploitation Velocity and AI-Accelerated Discovery

A significant contextual aspect of CVE-2026-31431's disclosure is the documented role of AI-assisted code analysis in its discovery. The Xint Code Research Team reported surfacing the vulnerability in approximately one hour of AI-directed scanning against the kernel's cryptographic subsystem — a subsystem that had remained vulnerable for nine years despite substantial human review [9]. This single data point does not support broad statistical claims; whether this represents a broader trend in AI-assisted vulnerability discovery remains to be established. The finding is notable in isolation regardless.

The practical implication for defenders is straightforward: the assumption that kernel-level logic bugs will take years to surface after introduction is no longer reliable. Microsoft Defender has reported observing preliminary testing activity and assessed that increased threat actor exploitation may follow in the near term [1], suggesting that exploitation campaigns may follow the public proof-of-concept release faster than historical baselines would predict. Organizations whose patch cadence for kernel updates is calibrated around multi-week remediation windows should treat Copy Fail as a trigger for reviewing that timeline.

Affected Scope

Linux kernels incorporating the 2017 in-place AEAD optimization through the versions immediately preceding vendor patches are affected. Given that this optimization was introduced in 2017, virtually all production Linux instances in active use today fall within the affected range. Ubuntu 26.04 (Resolute) and later distributions are not affected, as they ship with a kernel that already incorporates the fix [3][4]. Ubuntu 24.04 LTS, Amazon Linux 2023, RHEL 9.x and 10.x, SUSE 15 and 16, Debian stable, and most other currently supported distributions are vulnerable until vendor-specific patches are applied.

Recommendations

Immediate Actions

Organizations should treat kernel patch application for CVE-2026-31431 as an emergency change. Patched kernels are available or in active release from Ubuntu, Red Hat, SUSE, Amazon, CloudLinux, and OVHcloud, and should be applied to all Linux hosts as soon as operationally feasible [3][5][14][16]. The highest remediation priority should go to Kubernetes nodes, CI/CD build runners, and shared-kernel compute environments hosting AI or multi-tenant workloads, because these combine low-trust code execution with the highest potential blast radius.

On systems where immediate patching is not possible — particularly RHEL-family hosts where the `algif_aead` module is compiled directly into the kernel and cannot be removed or prevented from loading via `modprobe.d` — the most broadly applicable interim protection is a custom seccomp policy that denies `AF_ALG` socket creation [2][7]. Because the exploit's first step requires opening an `AF_ALG` socket, this filter prevents exploitation entirely on unpatched kernels without affecting the vast majority of workloads, which do not use the kernel's userspace crypto API directly. On distributions where `algif_aead` is a loadable module rather than a built-in, adding the module to the `modprobe.d` blacklist and rebooting provides an alternative workaround, though this approach does not function on RHEL-family systems with the module compiled in [14][2].

Security operations teams should also deploy or tune runtime detection rules targeting unexpected `AF_ALG` socket creation events. In Falco or equivalent eBPF-based runtime security tools, a rule alerting on `socket(AF_ALG, ...)` calls from non-cryptographic processes or from any containerized workload provides an early indicator of potential exploitation attempts, covering both the unpatched interim period and ongoing detection after patching [6].

Short-Term Mitigations

In the period following patch application, security teams should audit seccomp profiles across containerized workloads and CI/CD pipelines. Because the Kubernetes `RuntimeDefault` profile does not block `AF_ALG` socket creation, organizations seeking defense-in-depth should deploy a custom seccomp policy that explicitly denies `socket` calls with the `AF_ALG` domain to all workloads that have no legitimate need for this capability, which in practice is nearly all application containers [13]

[7]. This hardening step has independent value beyond Copy Fail: reducing the exploitable kernel attack surface available to containerized workloads is a sound posture regardless of the specific vulnerability driving the change.

Container runtime configurations should be reviewed to minimize the sensitive material available to any workload that might be compromised. Secrets management practices — including short-lived credentials, workload identity federation, and just-in-time secret access — limit the value of credentials accessible from a compromised node, reducing the downstream impact of privilege escalation even when the initial escalation cannot be prevented. Organizations using static, long-lived Kubernetes service account tokens should evaluate migration to projected volume tokens with bounded lifetimes as a near-term priority.

Incident response runbooks should be updated to reflect the in-memory-only artifact profile of Copy Fail exploitation. Standard digital forensics workflows focused on disk-based indicators of compromise will find no filesystem artifacts from the privilege escalation itself, as the page cache modification is in-memory only; investigators should therefore focus on network telemetry, memory images from live systems, and Kubernetes API audit logs, which persist across reboots and may capture post-escalation activity.

Strategic Considerations

Copy Fail concretizes a structural tension that has existed in cloud infrastructure design for years: shared-kernel multi-tenancy provides strong economic efficiency but weak isolation guarantees at the kernel level. Sandboxed container runtimes — including gVisor, Kata Containers, and AWS Firecracker-based solutions — interpose a separate kernel layer between container workloads and the host kernel, so that a privilege escalation in the container's virtualized kernel does not translate directly into host compromise [1][6]. Organizations running sensitive AI workloads, customer data processing, or genuinely multi-tenant services should evaluate whether their threat model justifies the operational overhead of sandbox runtimes on a subset of their node pools. If the trajectory suggested by Copy Fail and similar recent disclosures continues, the cost-benefit calculus for sandbox runtimes may shift materially.

The AI-accelerated discovery of Copy Fail also has implications for how organizations should think about vulnerability risk in mature, heavily reviewed codebases. Kernel code is among the most scrutinized software in existence, yet this bug persisted for nine years. If AI-assisted analysis can surface it in an hour, the same tools are available to adversaries scanning for unreported vulnerabilities. Organizations whose security investment is concentrated on application-layer security should ensure that infrastructure-layer patching — kernel updates, hypervisor updates, firmware — is given commensurate

urgency and automation. Automated kernel update pipelines with configurable maintenance windows are a reasonable response to a landscape in which, by several accounts, the mean time between vulnerability introduction and adversary exploitation appears to be compressing.

CSA Resource Alignment

Copy Fail maps directly to several threat categories and control domains addressed by CSA's published frameworks for cloud and AI security.

CSA's Cloud Threat Modeling 2025 identifies privilege escalation and container escape as among the highest-consequence threat paths in cloud environments, and explicitly notes that shared-kernel architectures require active mitigations rather than assumed isolation. The attack chain demonstrated by Copy Fail — an unprivileged attacker leveraging a kernel logic flaw to escalate through container boundaries to Kubernetes node root — is a concrete instance of the lateral movement and privilege escalation scenarios this framework describes. Cloud Threat Modeling 2025 provides a structured methodology for enumerating these risks within an organization's specific architecture, and security teams should use it to assess which node pools and workloads carry the highest aggregate exposure.

The AI Controls Matrix (AICM), CSA's framework integrating AI-specific risk into the Cloud Controls Matrix, addresses the security of AI compute infrastructure as a distinct risk domain. AI workloads running on shared Kubernetes nodes — including large language model inference services, training pipeline runners, and agentic AI orchestrators — are directly in scope for Copy Fail exposure: model weights, training datasets, API credentials, and inference outputs may all be accessible to an attacker who achieves node root. Organizations applying AICM controls should categorize kernel patch remediation as a mandatory infrastructure hardening activity, particularly under the controls governing AI system integrity and compute isolation.

CSA's Zero Trust guidance and the Certificate of Competence in Zero Trust (CCZT) curriculum both emphasize that trust cannot be derived from network position or container boundaries in isolation. Copy Fail reinforces this principle at the kernel level: a container that appears isolated from neighboring workloads at the network and namespace layers may share a kernel page cache with every other process on its host. Zero Trust implementations should incorporate kernel-level isolation verification — including validated seccomp profiles, runtime integrity monitoring, and evaluated sandbox runtime options — as part of their workload isolation attestation.

CSA's Managing Privileged Access in a Cloud-First World publication is directly applicable to limiting post-exploitation blast radius. Organizations that have implemented just-in-time privileged access, short-lived credentials, and workload identity controls constrain the lateral movement available to an attacker who achieves node root, even when the initial privilege escalation cannot be prevented before patching is complete.

Finally, MAESTRO — CSA's threat modeling framework for agentic AI systems — identifies substrate compromise as a risk vector for agentic workloads deployed on shared cloud infrastructure. An autonomous AI agent running on a shared Kubernetes node has the same exposure to Copy Fail as any other containerized process, but may additionally be positioned to exfiltrate particularly sensitive data — system prompts, conversation history, tool credentials — that aggregates into a high-value intelligence target. For organizations deploying agentic AI at scale, Copy Fail is a concrete instance of the MAESTRO substrate threat model and underscores the importance of infrastructure hardening as a prerequisite for safe agentic AI deployment.

References

[1] Microsoft Security Blog. "CVE-2026-31431: Copy Fail vulnerability enables Linux root privilege escalation across cloud environments." Microsoft, May 1, 2026.

[2] CERT-EU. "High Vulnerability in the Linux Kernel ('Copy Fail')." CERT-EU Security Advisory 2026-005, April 2026.

[3] Ubuntu Security Team. "Fixes available for CVE-2026-31431 (Copy Fail) Linux Kernel Local Privilege Escalation Vulnerability." Ubuntu Blog, 2026.

[4] Help Net Security. "Nine-year-old Linux kernel flaw enables reliable local privilege escalation (CVE-2026-31431)." Help Net Security, April 30, 2026.

[5] Wiz Research. "Copy Fail: Universal Linux Local Privilege Escalation Vulnerability." Wiz Blog, 2026.

[6] Sysdig. "CVE-2026-31431: 'Copy Fail' Linux kernel flaw lets local users gain root in seconds." Sysdig Blog, 2026.

[7] Tenable. "Copy Fail (CVE-2026-31431): Linux Kernel Privilege Escalation FAQ." Tenable Blog, 2026.

[8] NIST National Vulnerability Database. "CVE-2026-31431 Detail." NVD, 2026.

[9] Xint Code Research Team. "Copy Fail: 732 Bytes to Root on Every Major Linux Distribution." Xint, 2026.

[10] Theori / Copy.Fail. "Copy Fail – CVE-2026-31431." Official vulnerability disclosure, April 2026.

[11] Bugcrowd. "What we know about Copy Fail (CVE-2026-31431)." Bugcrowd Blog, 2026.

[12] University of Toronto Information Security. "'Copy Fail' Linux kernel LPE and container escape." University of Toronto, 2026.

[13] Juliet Security. "Copy Fail in Kubernetes: RuntimeDefault Did Not Block AF_ALG." Juliet.sh, 2026.

[14] CloudLinux. "CVE-2026-31431 (Copy Fail): Mitigation and Upcoming Patches for CloudLinux." CloudLinux Blog, 2026.

[15] Theori. "copy-fail-CVE-2026-31431 (GitHub)." GitHub, 2026.

[16] OVHcloud. "Copy.Fail (CVE-2026-31431): How to Rapidly Protect OVHcloud MKS Clusters from the Linux Kernel Zero-Day." OVHcloud Blog, 2026.