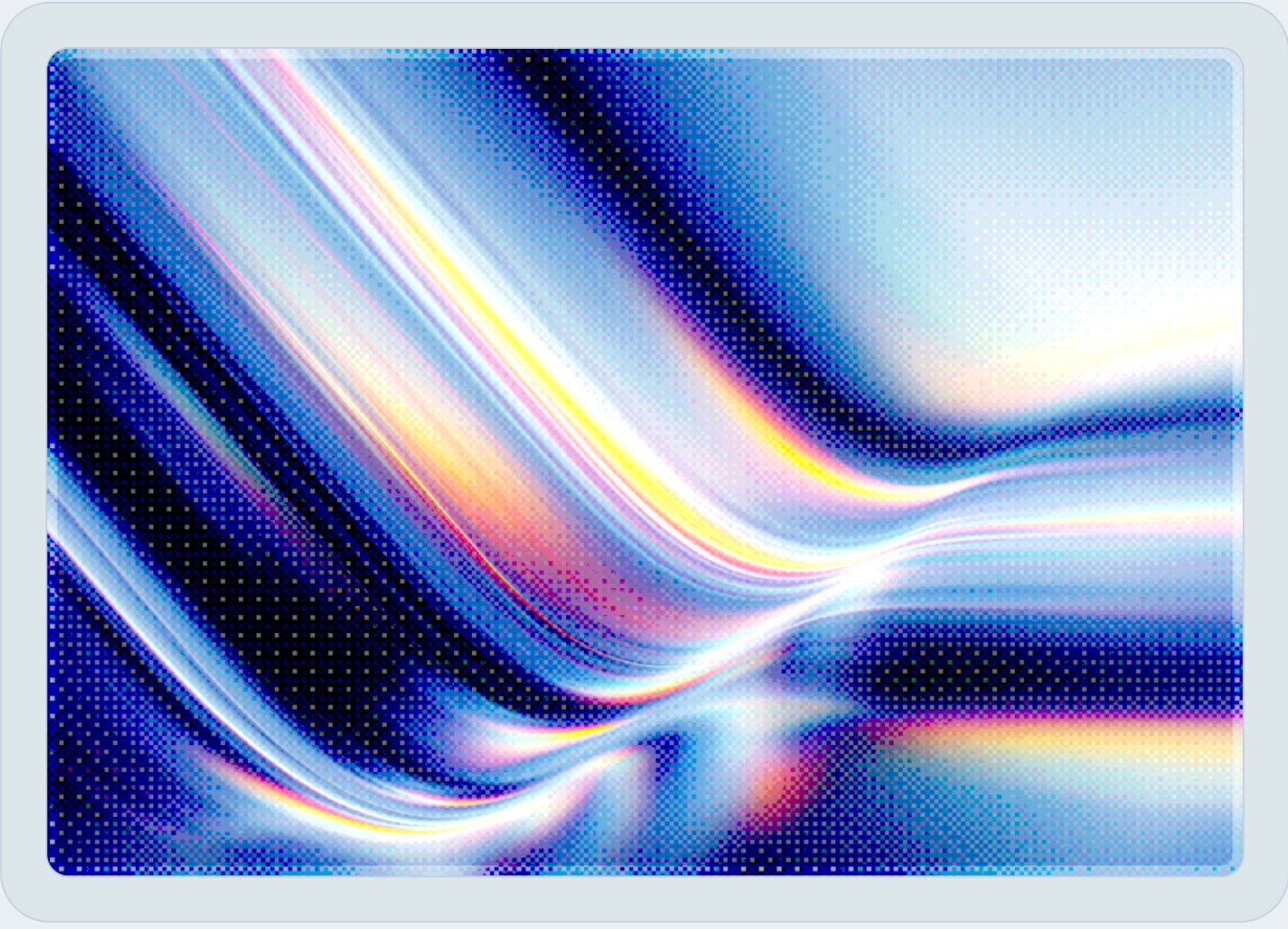
CSAI Foundation | Cloud Security Alliance

Linux DirtyFrag Wave: FIM Evasion on AI Infrastructure

How Page-Cache LPE Vulnerabilities Achieve Root Without Touching Disk

2026-06-28

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- A family of related Linux kernel local privilege escalation (LPE) vulnerabilities — DirtyFrag (CVE-2026-43284, CVE-2026-43500), Fragnesia (CVE-2026-46300), pedit COW (CVE-2026-46331), and DirtyClone (CVE-2026-43503) — share a root cause in the Linux page-cache architecture and have arrived in rapid succession from April through June 2026.
- DirtyFrag has been under active exploitation since at least May 8, 2026, with a public proof-of-concept available [1]; pedit COW and DirtyClone PoCs were published in late June 2026 [2][3].
- pedit COW and DirtyClone both achieve root by poisoning the in-memory copy of a setuid binary (/bin/su) without ever writing to disk. Standard file integrity monitoring (FIM) tools that compare on-disk hashes see nothing — attacks leave no trace in standard kernel audit logs.
- For threat actors seeking proprietary model access or persistent infrastructure access, AI/ML infrastructure represents among the highest-value targets: inference servers, GPU compute clusters, and multi-tenant Kubernetes nodes almost universally run affected Linux distributions and typically lack the runtime memory-monitoring controls needed to detect this attack class.
- All affected distributions have patches available; operators should treat this as a critical patching emergency — given active exploitation and available public PoCs — and supplement patching with runtime detection and Kubernetes seccomp hardening.

Background

The Linux Page-Cache Architecture as an Attack Surface

Linux uses a page cache — a region of kernel memory that holds the contents of on-disk files — to avoid redundant disk reads and to share file-backed memory efficiently across processes. When a process reads an executable, the kernel maps that file's pages from the page cache, and multiple processes may share the same physical page. Kernel subsystems that operate on in-memory data, such as cryptographic decryption and packet processing, are expected to create private copies of those pages

before writing to them, a mechanism known as copy-on-write (COW). The DirtyFrag vulnerability family exploits a recurring failure in that expectation: bugs in which networking or packet-processing code writes into a shared page-cache page that it never made private, thereby corrupting the in-memory representation of a file — including setuid executables — without touching the underlying disk storage.

This fundamental architecture has produced exploitable flaws before. Dirty Pipe (CVE-2022-0847) demonstrated in 2022 that the page cache could be corrupted through the splice() syscall [4]. The 2026 wave begins with Copy Fail (CVE-2026-31431), disclosed in April, which exploited a similar path in the xfrm subsystem [5]. What followed was a cascade of related disclosures, suggesting that security researchers were systematically probing adjacent code paths for analogous defects.

Disclosure Timeline

The 2026 wave of page-cache LPE vulnerabilities unfolded over roughly three months. Copy Fail (CVE-2026-31431) emerged in April as the first member of the family, exploiting the xfrm-ESP (IPsec) subsystem. DirtyFrag (CVE-2026-43284, CVE-2026-43500) was disclosed on May 7, 2026, when an embargo was broken by an unrelated third party; within 24 hours Microsoft confirmed active exploitation in the wild [1]. A third member, Fragnesia (CVE-2026-46300), was disclosed alongside DirtyFrag as a related networking-path flaw [6]. Then in late June, pedit COW (CVE-2026-46331) was disclosed via The Hacker News on June 26 [2], and JFrog Security Research published a working exploit chain for DirtyClone (CVE-2026-43503) on June 25 [3]. All five CVEs share the same page-cache write-primitive structure and lead to the same outcome: unprivileged local root.

Security Analysis

The DirtyFrag Family: Shared Architecture, Different Entry Points

Although the five vulnerabilities carry distinct CVEs and exploit different kernel subsystems, they share a common exploitation template. Each finds a code path where the kernel performs an in-place write into a page-cache–backed page — one the kernel should have first privatized via COW — allowing an attacker to overwrite the cached content of a file. Because the Linux page cache is the authoritative in-memory representation of file contents, any process that subsequently reads or executes from that file, including privileged processes and the kernel itself, observes the corrupted version. Copy Fail (CVE-2026-31431) and Fragnesia (CVE-2026-46300) both exploit this primitive through distinct networking entry points; the analysis below focuses on the three members with published proof-of-concept exploit chains.

DirtyFrag (CVE-2026-43284) targets the xfrm-ESP IPsec receive path [7]. The bug, present since 2017 [8], allows the ESP decryption function to skip the skb_cow_data() call for specific non-linear socket buffer configurations and write four bytes of attacker-controlled data into a page-cache page via a spliced pipe [8]. CVE-2026-43500 provides an analogous path through the RxRPC subsystem, which supports the Andrew File System protocol, broadening the attack surface. Both CVEs carry CVSS 3.1 scores of 8.8 and 7.8 respectively [6].

pedit COW (CVE-2026-46331) reaches the page cache through an entirely different entry point: the traffic-control packet editing subsystem (act_pedit). The function tcf_pedit_act() computes its copy-on-write range before applying typed-key offsets, producing a mismatch between the range it privatized and the range it writes [2][9]. The result is an out-of-bounds write into a shared page-cache page without a private copy, allowing an unprivileged user to corrupt the cached image of /bin/su and gain root upon execution.

DirtyClone (CVE-2026-43503) exploits a missing flag propagation in __pskb_copy_fclone() [3]. When a netfilter TEE rule clones a packet, the clone does not inherit the SKBFL_SHARED_FRAG flag. The resulting unflagged skb then passes through the IPsec in-place decryption path — which treats the absence of the flag as permission to write directly — overwriting the file-backed page-cache page backing /usr/bin/su [10]. JFrog Security Research's June 25 publication of a working exploit chain confirmed this is not a theoretical concern [3].

Fragnesia (CVE-2026-46300) was disclosed alongside DirtyFrag and shares the core exploitation template — a page-cache write into an uncopied shared page — but reaches that page through a distinct networking entry point. Its separate CVE reflects a different entry point rather than a qualitatively different attack primitive; readers are referred to [6][13] for the full technical analysis.

File Integrity Monitoring Evasion

The FIM evasion characteristic shared by pedit COW and DirtyClone deserves particular attention, because it upends a standard assumption about endpoint detection. Most deployed FIM solutions — including those embedded in endpoint detection and response (EDR) products — operate by computing cryptographic hashes of files on disk and comparing them against a known-good baseline. Some also monitor inotify events, which the kernel fires when file contents are modified at the filesystem layer. Neither mechanism observes changes to the kernel's in-memory page cache that do not result in a disk write.

Both pedit COW and DirtyClone exploit precisely this gap. The attack corrupts the cached in-memory copy of a setuid binary, typically /bin/su or /usr/bin/su, by injecting a small shellcode payload. When the victim process executes that binary, it executes from the poisoned in-memory cache rather than reading

from disk. The file on disk is never modified; no inotify event fires; no hash comparison detects a change. FIM reports system files as unmodified while a root shell is already open [2][3]. Neither exploit leaves entries in standard kernel audit logs, further narrowing the available detection surface.

This FIM evasion is not incidental — it follows structurally from exploiting the page cache rather than the filesystem, meaning any attack in this class will share the same evasion property. It suggests that organizations relying solely on disk-based integrity monitoring as their primary detection control for host compromise are exposed to an entire class of attacks that this tooling was never designed to catch.

AI Infrastructure Exposure

AI and machine learning infrastructure faces compounded risk from this vulnerability family for several converging reasons.

AI training and inference workloads on cloud infrastructure run almost exclusively on Linux, particularly on GPU compute instances from major providers [6][7]. Inference servers, model training clusters, and GPU compute nodes across Amazon EC2, Google Cloud, and Microsoft Azure run Ubuntu, Red Hat Enterprise Linux, or Debian-derivative distributions, all of which were affected at disclosure time. The breadth of this Linux footprint means organizations cannot rely on architectural heterogeneity to limit blast radius.

Multi-tenant Kubernetes deployments create a container escape vector. Research has demonstrated that an unprivileged Kubernetes pod can exploit DirtyFrag (CVE-2026-43284) to achieve node-level code execution on Amazon EKS by corrupting binaries in shared container image layers [11]. Because multiple pods on the same node share page-cache–backed copies of image-layer files, a corruption introduced by one pod's process affects every other container reading from the same cached copy. Kubernetes runs privileged infrastructure components as DaemonSets on every node; a DaemonSet that executes a corrupted binary with elevated privileges achieves full node compromise. GPU nodes, which are often the highest-value targets in an AI cluster, are as vulnerable as any other node type.

AI inference servers and training jobs frequently run under scheduling frameworks and orchestration layers that assume kernel-level isolation between workloads. When an LPE vulnerability breaks that assumption, an attacker who achieves initial access to a low-privilege inference worker — for example, through a prompt injection that triggers shell execution, or through a vulnerability in the model-serving framework itself — can escalate to node root, extract model weights, inject poisoned inference results for co-resident workloads, or pivot to control-plane infrastructure. The concentration of high-value AI assets on GPU nodes — model weights, training data, API credentials — makes this escalation path a logical target for sophisticated threat actors seeking intellectual property or persistent infrastructure access.

AI infrastructure operators frequently face operational pressure to defer kernel patching on GPU nodes, where reboots interrupt long-running training jobs. In practice, this means nodes may be explicitly excluded from routine patching windows, creating exposure gaps after patches become available. For vulnerabilities with available public exploits and active in-the-wild exploitation, this patching lag translates directly into extended exposure windows on the highest-value nodes in an AI deployment.

Recommendations

Immediate Actions

Patch affected kernels without delay. All major distributions have issued updated kernels addressing the DirtyFrag family CVEs. For DirtyFrag (CVE-2026-43284, CVE-2026-43500), patches have been available since May 2026 across Ubuntu, RHEL 8/9/10, Debian, Fedora, AlmaLinux, and CentOS Stream [6][7][12]. Patches for pedit COW (CVE-2026-46331) and DirtyClone (CVE-2026-43503) have been available in mainline since May 21, 2026, with distribution backports following through June [9][12]. Note that mainline patches for pedit COW and DirtyClone were committed during the pre-disclosure coordinated embargo period; distribution-specific backports were released through June following the embargo lift on approximately June 26, 2026. Operators should verify running kernel versions against their distribution's patched releases and apply updates immediately. For AI infrastructure operators unable to immediately reboot GPU nodes, temporary module-level mitigations are available.

Apply module blacklist mitigations for unpatched systems. Until a patched kernel can be deployed, the attack surface for DirtyFrag can be reduced by blacklisting the vulnerable kernel modules. Blacklisting xfrm_algo, esp4, esp6, and rxrpc prevents the ESP and RxRPC attack paths from being exercised. However, this will break functionality for IPsec VPN deployments using strongSwan or similar tools, and for any AFS consumers using RxRPC. Operators must assess their network architecture before applying these mitigations [6][8].

Audit Kubernetes seccomp profiles. Kubernetes clusters without seccomp enforcement — which represent a significant portion of production deployments — allow the syscalls required to initiate the DirtyFrag chain [11]. Enabling the RuntimeDefault seccomp profile — or a custom profile that blocks unprivileged AF_KEY, AF_RXRPC, and XFRM socket operations — substantially raises the exploitation barrier in containerized environments. EKS, GKE, and AKS all provide mechanisms to enforce seccomp profiles cluster-wide; operators should verify these are active [11][13].

Short-Term Mitigations

Deploy runtime detection for page-cache write primitives. Because FIM provides no signal against this attack class, defenders must supplement or replace it with controls that observe kernel memory operations. Runtime security tools based on eBPF — including Falco, Tetragon, and similar frameworks — can be configured to alert on the specific syscall sequences and socket operations characteristic of DirtyFrag exploitation, including unprivileged use of AF_KEY socket creation, XFRM netlink operations, and splice() into named file descriptors [8][14]. These detections are imperfect but provide substantially more coverage than disk-based FIM alone.

Extend FIM posture with memory-integrity monitoring. Organizations should evaluate whether their endpoint security tooling includes any capability to detect in-memory file corruption. This is a capability gap in most traditional FIM products. Security teams should request confirmation from their EDR vendors about detection coverage for CVE-2026-46331 and CVE-2026-43503 specifically, and should not assume that positive FIM status implies a system is uncompromised.

Enforce least-privilege networking policies on AI workloads. The DirtyFrag family exploits kernel networking subsystems — IPsec, RxRPC, traffic control, and netfilter — that most AI inference and training workloads do not legitimately use. Network policies in Kubernetes that restrict pod-level access to these subsystems, combined with AppArmor or SELinux profiles that block the relevant syscalls at the process level, reduce the attack surface available to a compromised workload.

Strategic Considerations

Reassess patching cadence for GPU and AI nodes. The operational reluctance to reboot GPU compute nodes — driven by concerns about interrupting long-running training jobs — creates a structural vulnerability in AI infrastructure patching programs. Organizations should develop kernel live-patching strategies using tools such as KernelCare, Livepatch, or kpatch, which allow kernel security patches to be applied without rebooting. The DirtyFrag wave demonstrates that this is not a theoretical risk: GPU nodes are high-value targets, and patch delays are operationally consequential.

Treat kernel LPE as an AI-model integrity risk. While AI security programs have extensively addressed model-layer attacks — adversarial inputs, training poisoning, and inference manipulation — host-level kernel security has received comparatively less attention in AI-specific threat modeling. The DirtyFrag family demonstrates that kernel-level compromise provides a path to the same outcomes: an attacker with root on an inference node can directly manipulate model weights in memory, inject malicious outputs, or exfiltrate proprietary models without triggering most model-layer detection controls. AI security governance should formally incorporate host-level kernel security into threat models for AI systems.

Evaluate multi-tenant isolation architecture. Organizations running shared GPU Kubernetes clusters for multiple AI teams or customers should assess whether tenant isolation assumptions hold under the container escape scenario demonstrated for CVE-2026-43284 [11]. Physical node-per-tenant separation or stronger isolation primitives such as gVisor or Kata Containers may be warranted for high-sensitivity AI workloads.

CSA Resource Alignment

The DirtyFrag vulnerability family and its implications for AI infrastructure map directly to several CSA frameworks and guidance documents.

CSA's MAESTRO framework (Multi-Agent Environment, Security Threats, Risk Operations) addresses the threat modeling of agentic AI deployments, including the host infrastructure on which AI agents execute [15]. The kernel LPE risk described in this note is a foundational MAESTRO concern: an attacker who escalates to root on an inference or agent node gains control over the agent's execution environment, its tool access, and its memory — all of which MAESTRO's threat model treats as high-severity compromise vectors.

CSA's AI Controls Matrix (AICM) — an AI-specific superset of the Cloud Controls Matrix (CCM) — provides control categories relevant to kernel patching, runtime integrity, and container isolation. The Vulnerability and Patch Management domain and the Infrastructure Security domain both apply directly: AICM controls in these areas address timely remediation of critical kernel vulnerabilities and cover runtime monitoring for host-level compromise [16]. Organizations using CSA's STAR program for AI system assurance should document their DirtyFrag remediation status as evidence of AICM conformance.

CSA's Zero Trust Guidance is also directly applicable. Zero Trust architectures do not assume that a host that was clean at boot time remains clean at runtime; they require continuous verification of workload integrity. The FIM-evasion characteristics of pedit COW and DirtyClone illustrate exactly why perimeter-and-boot-time trust models fail: an attacker who poisons the page cache after boot is invisible to controls that only verify state at provisioning time [17]. Zero Trust implementations for AI infrastructure should incorporate continuous runtime attestation and behavioral anomaly detection, not solely disk-state integrity checks.

The CSA AI Organizational Responsibilities guidance addresses the governance and accountability structure that AI operators must maintain [18]. In the context of this vulnerability wave, that includes assigning clear ownership for AI infrastructure patching cadence, ensuring kernel security is within scope

for AI security reviews, and requiring vendors of AI platforms deployed on Linux to document their patch applicability and distribution timelines.

References

[1] Microsoft Security. "Active attack: Dirty Frag Linux vulnerability expands post-compromise risk." Microsoft Security Blog, May 8, 2026.

[2] The Hacker News. "New Linux pedit COW Exploit Enables Root Access by Poisoning Cached Binaries." The Hacker News, June 26, 2026.

[3] JFrog Security Research. "Dissecting and Exploiting Linux LPE Variant: DirtyClone (CVE-2026-43503)." JFrog, June 25, 2026.

[4] Max Kellermann. "Dirty Pipe (CVE-2022-0847)." cm4all.com, March 2022.

[5] ReversingLabs. "How Dirty Frag rose from the Copy Fail exploit." ReversingLabs Blog, May 12, 2026.

[6] Tenable. "Dirty Frag (CVE-2026-43284,CVE-2026-43500): Linux Kernel Privilege Escalation FAQ." Tenable Blog, May 8, 2026.

[7] Wiz. "Dirty Frag (CVE-2026-43284) Linux Privilege Escalation." Wiz Blog, May 8, 2026.

[8] Qualys. "Dirty Frag Linux LPE Zero-Day: Detect & Mitigate Now." Qualys Blog, May 9, 2026.

[9] TuxCare. "pedit-cow (CVE-2026-46331): Linux tc Flaw Grants Root." TuxCare Blog, June 25, 2026.

[10] Threat-Modeling.com. "CVE-2026-43503: 'DirtyClone' Linux Kernel Local Privilege Escalation to Root via Cloned Network Packets." Threat-Modeling.com, June 27, 2026.

[11] GitHub / Percivalll. "Dirty-Frag-Kubernetes-PoC: PoC demonstrating how an unprivileged Kubernetes Pod can achieve node-level code execution on EKS via CVE-2026-43284." GitHub, 2026.

[12] Ubuntu Security. "Dirty Frag Linux kernel local privilege escalation vulnerability mitigations." Ubuntu Blog, May 8, 2026.

[13] Red Hat Customer Portal. "RHSB-2026-003 Networking subsystem Privilege Escalation - Linux Kernel (CVE-2026-43284, CVE-2026-43500, CVE-2026-46300) - Dirty Frag." Red Hat, 2026.

[14] Sysdig. "Dirty Frag: Detecting unpatched local privilege escalation via Linux Kernel ESP and RxRPC." Sysdig Blog, May 8, 2026.

[15] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[16] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA, 2025.

[17] Cloud Security Alliance. "Zero Trust Working Group." CSA, 2024.

[18] Cloud Security Alliance. "AI Organizational Responsibilities: Governance, Risk Management, Compliance and Cultural Aspects." CSA, 2024.