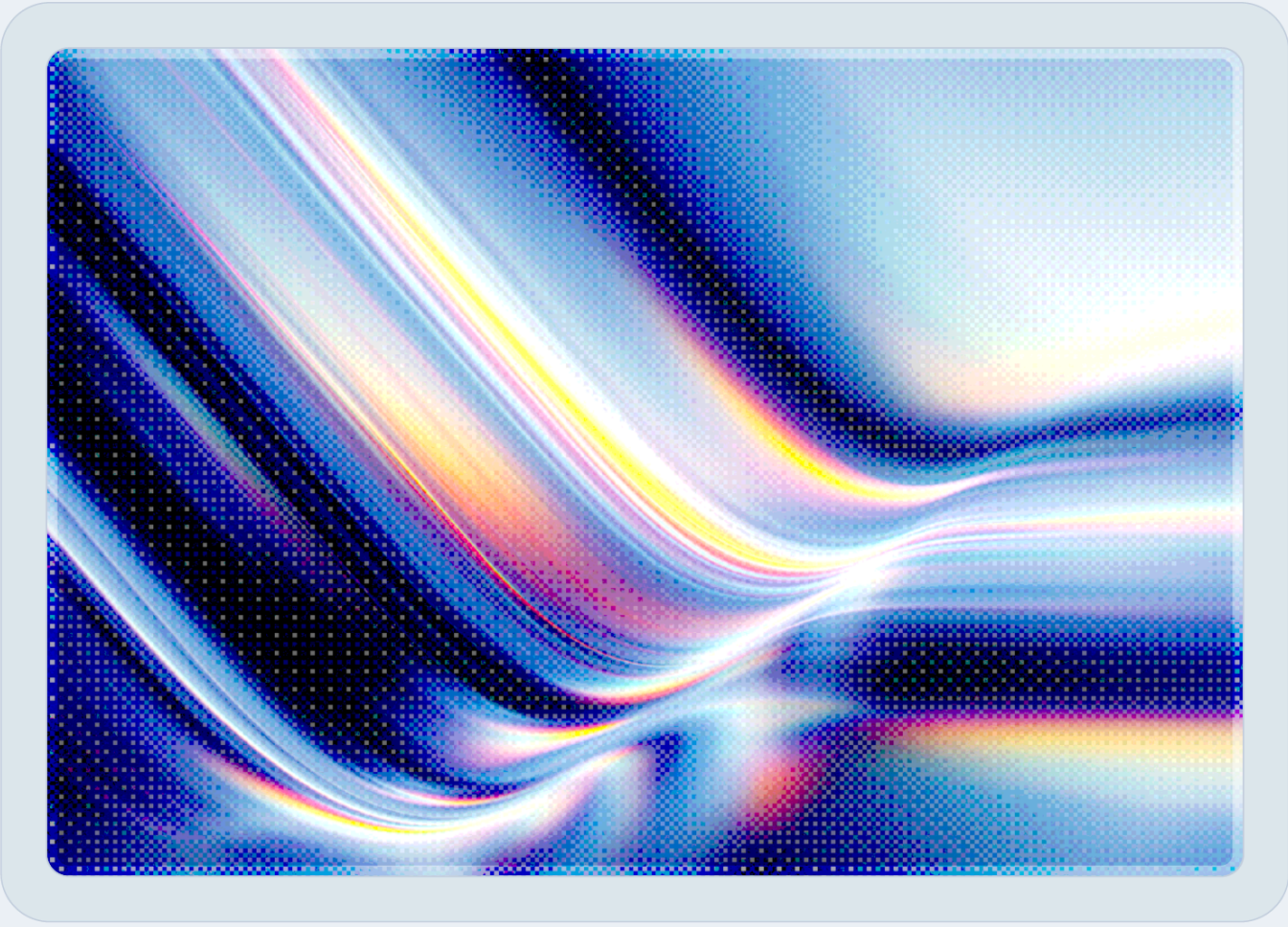
CSAI Foundation | Cloud Security Alliance

Dirty Frag: Linux LPE Threatens AI Container Workloads

CVE-2026-43284 and CVE-2026-43500 — A Partially Unpatched Kernel Privilege Escalation Chain

2026-05-09

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Dirty Frag chains two page-cache write primitives — CVE-2026-43284 in the xfrm-ESP (IPsec) subsystem and CVE-2026-43500 in the RxRPC protocol implementation — into a deterministic local privilege escalation that requires no race condition and delivers host root access with a single command on all major Linux distributions [1][4].
- A public proof-of-concept has been circulating since May 7, 2026; Microsoft Threat Intelligence reported limited in-the-wild active exploitation the following day [5][15].
- As of May 9, 2026, CVE-2026-43284 has been patched in the mainline kernel and in select distribution releases; CVE-2026-43500 has no upstream patch available, leaving all systems with the rxrpc module loaded in a state of ongoing exposure [10][14].
- AI inference and training workloads face elevated risk because containers share the host kernel's page cache. In default Kubernetes configurations (PSS Unset or Privileged baseline), containers can access AF_KEY, XFRM netlink, and AF_RXRPC sockets without explicit privilege grants. Default Docker deployments block AF_RXRPC via the built-in seccomp profile but do not restrict AF_KEY or XFRM netlink socket creation, leaving Docker-hosted AI workloads on unpatched hosts exposed to the xfrm-ESP attack path [4][16].
- Immediate mitigations include applying available kernel updates, blacklisting the esp4, esp6, and rxrpc modules where operationally safe, and deploying seccomp profiles that restrict the relevant socket families across AI container workloads [11][13].

Background

Dirty Frag is the second broad-scope Linux local privilege escalation vulnerability disclosed in the span of eight days. On April 30, 2026, researcher Hyunwoo Kim submitted a responsible-disclosure report to Linux kernel maintainers describing two separate page-cache write vulnerabilities: one in the kernel's xfrm-ESP (IPsec Encapsulating Security Payload) subsystem and one in the RxRPC protocol implementation. On May 7, before distributions had completed their coordinated patch rollout, an embargo break forced early public disclosure, and a working proof-of-concept was made available

simultaneously — leaving systems running kernels with the esp4 and esp6 modules loaded, a configuration present in virtually all production Linux kernels since at least 2017, exposed before mitigations were in place [1][2][4].

The vulnerability shares architectural roots with Dirty Pipe (CVE-2022-0847) and the recently disclosed Copy Fail (CVE-2026-31431). Like those predecessors, Dirty Frag exploits the Linux kernel's page-cache model — the in-memory representation of filesystem contents — to corrupt data that an unprivileged process should never be able to modify. The name reflects the specific attack surface: the frag member of the kernel's struct sk_buff (socket buffer), the data structure used to pass network data through the kernel stack. By manipulating how fragmented socket buffers interact with in-place cryptographic decryption, the attacker achieves a controlled write into the page cache, which can be directed at any readable file on the system, including privileged binaries such as /usr/bin/su [3][4].

The disclosure arrived under particularly difficult circumstances for Linux distribution security teams. Copy Fail (CVE-2026-31431), disclosed on April 29, 2026, had already triggered a simultaneous kernel update effort across the major distributions. The Dirty Frag embargo break on May 7 forced those same teams to pivot to a second critical LPE before Copy Fail remediation was complete — and critically, Copy Fail mitigations provide no protection against Dirty Frag, as the two vulnerabilities target entirely different kernel subsystems [5][6]. For operators who had not yet completed Copy Fail remediation when the Dirty Frag embargo broke, kernel vulnerability exposure was effectively continuous across the ten days following Copy Fail's initial disclosure.

Security Analysis

The Page-Cache Write Primitive

The root cause of both CVEs lies in the in-place decryption fast paths of the esp4 and esp6 modules (CVE-2026-43284) and the rxrpc module (CVE-2026-43500). Under normal operation, when the kernel decrypts incoming IPsec-protected or RxRPC data, it decrypts directly over the memory buffers containing the incoming packet data. This approach avoids a redundant data copy and is generally safe — but only when the kernel has exclusive ownership of those memory buffers [3][7].

The vulnerability emerges when an attacker provides socket buffers that are not exclusively kernel-owned. By using the splice(2) or sendfile(2) system calls, or the MSG_SPLICE_PAGES socket option, an unprivileged process can insert a reference to a read-only page-cache page into the frag slot of a sender-side socket buffer. A natural target is a cached page of a privileged binary: /usr/bin/su, /etc/passwd, or a shared library. When the receiving kernel path later decrypts the incoming data, it

performs the cryptographic write directly over that externally-backed page, modifying the in-RAM representation of the target file without touching the on-disk copy. The result is a controlled 4-byte write anywhere within a cached page of any readable file's in-memory representation, modifying that page without altering the corresponding on-disk copy [3][4].

The ESP flaw has been present in the kernel since at least 2017, meaning virtually every production Linux kernel in active use today contains the vulnerable code path [4]. The RxRPC flaw was introduced more recently, in 2023, limiting its exposure to systems running kernels from the past three years [3]. Unlike timing-dependent race-condition exploits — which may fail, require many attempts, or trigger kernel panics on unsuccessful runs — this is a deterministic logic flaw. The exploit either succeeds or quietly fails; there is no observable crash or system instability to alert defenders [4][8].

The publicly available proof-of-concept (github.com/V4bel/dirtyfrag) has been confirmed to deliver root access in a single command on Ubuntu 24.04.4, RHEL 10.1, openSUSE Tumbleweed, CentOS Stream 10, AlmaLinux 10, and Fedora 44 [1][2][9]. This breadth of affected distributions reflects the universal nature of a bug rooted in widely enabled kernel modules rather than distribution-specific packaging choices.

Patch Status and Active Exploitation

The vulnerability's public exposure arrived before any distribution had completed patch delivery. The Linux Kernel Organization published a fix for CVE-2026-43284 (the xfrm-ESP half) on May 8, 2026. AlmaLinux and CloudLinux released patched kernel packages within hours [10][13]; Ubuntu and Canonical published module-blacklist mitigations describing interim exposure controls while full kernel patch packages were being prepared [11][12]. Red Hat assigned CVE-2026-43284 an "Important" severity rating and is expediting delivery for RHEL subscribers [14]. CVE-2026-43500 (the RxRPC half), however, has no upstream patch available as of this writing. Organizations that cannot immediately apply the CVE-2026-43284 kernel update, and any organization running a kernel that ships the rxrpc module, face a period of elevated and partially unmitigated exposure.

Microsoft's Threat Intelligence Center reported observing limited in-the-wild exploitation activity as of May 8, 2026. The observed pattern involves SSH-based initial access, followed by execution of a staged ELF binary that performs privilege escalation via su, and subsequent reconnaissance and modification of application configuration files — consistent with post-compromise lateral movement [15]. The attack chain was observed in an enterprise context unrelated to AI workloads, but the underlying privilege escalation technique is not workload-specific and applies equally to AI and non-AI Linux infrastructure. The availability of a public, single-command proof-of-concept means that opportunistic and automated

exploitation can be expected to spread to cloud and AI workloads rapidly — a pattern consistent with the speed at which prior Linux LPE disclosures with readily available proofs-of-concept have been weaponized.

AI Container Workload Risk

The risks posed by Dirty Frag carry specific and heightened implications for AI inference and training infrastructure. Modern AI deployments rely heavily on containerized environments — Kubernetes clusters, Docker-based inference services, GPU-accelerated multi-tenant compute platforms — almost all of which run on Linux hosts sharing a common kernel. These containers share the host kernel's page cache by design; isolation is achieved through namespaces and control groups, not through kernel memory separation. A page-cache write primitive therefore crosses container boundaries by nature: an attacker who corrupts a cached page inside one container can affect files used by the host or by other containers on the same node.

In a default Docker or containerd deployment without explicit hardening, containers can create AF_KEY and XFRM netlink sockets by default, providing the necessary prerequisites for the Dirty Frag xfrm-ESP exploit chain. The exploit additionally requires user namespace creation, which is also permitted by default in most Linux distributions. Security researchers tested this exposure in production managed Kubernetes environments and found that Amazon EKS clusters running Amazon Linux 2023 and Google Kubernetes Engine clusters running Container-Optimized OS were exploitable when Pod Security Standards were left at their default "Unset" or "Privileged" baselines. The Kubernetes Restricted pod security profile blocked the attack prerequisites in testing on both platforms [16].

The critical implication for AI security is that an attacker who gains code execution inside an AI inference container — whether through a model serving vulnerability, a supply chain compromise, an exposed API endpoint, or a vulnerable third-party dependency — can use Dirty Frag to escalate from container-local code execution to host root. From host root, the scope of compromise expands dramatically: all co-resident containers and their data become accessible, cloud provider instance metadata service (IMDS) endpoints can be queried for IAM credentials, container images and volumes can be modified, and the compromised node can be used as a pivot point to adjacent compute infrastructure. In multi-tenant AI inference platforms, a successful exploit grants an attacker access to co-resident workloads' data and resources, effectively eliminating container-level workload isolation on the affected node.

AI workloads introduce several compounding risk factors beyond those affecting general server infrastructure. GPU-accelerated containers are often configured with elevated privileges to facilitate CUDA device access or hardware memory management, providing the capability grants that lower the exploitation barrier. Training workloads that communicate across nodes via high-bandwidth fabrics (InfiniBand, RoCE) may have IPsec or XFRM-related modules loaded to support encrypted inter-node

data transfer, directly enabling the CVE-2026-43284 attack path. Organizations running multi-tenant inference services — where one customer's container and another's share a host — face the highest aggregate risk, because those services are simultaneously the most attractive exploitation target (access to one container offers a foothold across all tenants) and often among the least hardened against host-escaping LPE techniques, given operational pressure to prioritize availability and performance over kernel security controls.

Recommendations

Immediate Actions

The highest-priority action for any organization running Linux-based AI workloads is to apply available kernel updates for CVE-2026-43284 without delay. Patched kernels are available from AlmaLinux [10] and CloudLinux [13] as of May 8–9, 2026; Ubuntu and Canonical have published module-blacklist mitigations [11][12] with full kernel patch packages in preparation. Red Hat [14] and SUSE patches are in active delivery. Distribution security advisories should be monitored continuously for CVE-2026-43500 patch availability, and that update should be applied as soon as it is released. Kernel updates will require a system reboot; organizations should plan coordinated maintenance windows for AI workload hosts and account for the operational cost of restarting long-running training jobs.

For systems that cannot be immediately patched, module-level mitigations can reduce exposure without a reboot. Where IPsec tunnels (strongSwan, Libreswan, or similar) are not in active use, blacklisting the esp4 and esp6 kernel modules removes the CVE-2026-43284 attack surface. Where RxRPC (used by the AFS distributed filesystem) is not required, blacklisting the rxrpc module prevents the CVE-2026-43500 path. These blacklists take effect immediately via modprobe deny-listing and do not require a kernel update, but they require the modules to not currently be loaded. Organizations that rely on IPsec for encrypted inter-node AI cluster communication must not apply the esp4/esp6 blacklist and should instead prioritize the kernel update as the primary mitigation path.

Security teams should immediately audit container runtime configurations across AI workload environments to determine whether containers can create XFRM netlink or AF_RXRPC sockets, and whether user namespace creation is permitted. Any container on an unpatched host with default seccomp settings should be treated as potentially exploitable.

Short-Term Mitigations

Beyond module blacklisting, deploying or updating seccomp profiles for AI container workloads provides defense-in-depth that remains valuable even after patching, since it reduces exposure to this entire class of socket-based kernel attack. A seccomp profile that denies creation of AF_RXRPC (address family 33) and AF_ALG (address family 38) sockets blocks the RxRPC variant, since the exploit requires both socket families in combination. AI container workload seccomp profiles should also restrict AF_KEY and XFRM netlink socket creation where those capabilities serve no operational purpose; closing these gaps provides meaningful defense-in-depth regardless of patch status [16].

Kubernetes administrators should enforce the Restricted Pod Security Standard across all AI workload namespaces that do not have a documented requirement for the exempted capabilities. Testing confirmed that Restricted PSS blocked the prerequisites for Dirty Frag exploitation on both EKS and GKE [16]. This standard is particularly actionable for inference serving infrastructure, where containers generally should not require privileged access, broad capability grants, or user namespace creation. Where workloads have legitimate requirements that conflict with Restricted PSS, a custom seccomp profile scoped to those requirements is preferable to operating at a higher privilege tier.

AI platform teams running multi-tenant inference services should revisit their workload isolation assumptions in light of this vulnerability class. Even with Dirty Frag patches applied, the architectural pattern of sharing a host kernel across mutually distrusting tenants creates a recurring exposure surface with each new kernel LPE disclosure. Teams should consider node-level tenant isolation for high-sensitivity workloads — separating different customers' inference containers onto dedicated Kubernetes nodes — rather than relying solely on namespace-based isolation. This approach trades some compute density for a substantially stronger security boundary.

Endpoint detection coverage should be verified to include Linux hosts running AI workloads. The in-the-wild exploitation pattern documented by Microsoft involves execution of a staged ELF binary followed by su-based privilege escalation [15]. Behavior-based detection rules for unusual privilege escalation sequences and post-exploitation reconnaissance activities — particularly on hosts running AI inference or training workloads — should be deployed and tuned.

Strategic Considerations

Dirty Frag is the second broad-scope Linux local privilege escalation vulnerability affecting all major distributions running kernels with the esp4, esp6, or rxrpc modules loaded — the default configuration for most Linux installations — disclosed in the span of eight days, and the third major page-cache write vulnerability in the Linux kernel in four years, following Dirty Pipe (2022) and Copy Fail (2026). This pattern suggests that the kernel's page-cache and splice interaction with socket buffer fragments is a

persistent source of security debt. Organizations that deploy AI workloads on shared Linux infrastructure should treat kernel hardening — including module restriction policies, mandatory access control, and container runtime seccomp profiles — as standing baseline requirements rather than incident-driven responses to individual disclosures.

The eight-day disclosure-to-exploitation window observed with Dirty Frag suggests that patch deployment cycles assuming weeks of lead time may no longer be realistic for critical Linux LPE disclosures — particularly given the ready availability of single-command public proofs-of-concept. Organizations managing large Linux fleets should evaluate kernel live-patching solutions (kpatch for RHEL, Canonical Livepatch for Ubuntu) that can reduce the remediation window without requiring workload restarts or scheduled maintenance windows.

For AI infrastructure specifically, the concurrent emergence of credential-targeting Linux threats alongside kernel LPE disclosures suggests that adversaries may increasingly compose multi-stage attack chains against AI deployments — initial access via developer credential theft followed by post-compromise expansion using kernel LPE techniques such as Dirty Frag. While no publicly confirmed campaign linking these techniques against AI workloads has been documented as of this writing, security programs for AI platforms should be evaluated for their ability to detect and disrupt both the initial-access and the privilege-escalation legs of this emerging attack pattern.

CSA Resource Alignment

Dirty Frag maps to several layers of CSA's AI security frameworks. The MAESTRO agentic AI threat modeling framework identifies the execution environment and host infrastructure as critical attack surfaces in multi-agent deployments, particularly at Layer 3 (Agent Frameworks) and Layer 4 (Deployment and Infrastructure) where agent workloads share kernel state with co-resident processes [17]. Dirty Frag is a concrete instantiation of the MAESTRO threat class in which host-level compromise of AI infrastructure collapses workload confidentiality, integrity, and isolation guarantees. Security architects designing AI agent deployment environments should incorporate MAESTRO-based threat modeling of host kernel LPE paths into their isolation architecture reviews.

The CSA AI Controls Matrix (AICM), which extends the Cloud Controls Matrix for AI-specific environments, provides the primary governance framework applicable to this threat [18]. The AICM's infrastructure security domain addresses compute isolation, host hardening, and workload separation requirements that Dirty Frag exploitation circumvents. Controls governing container runtime hardening, kernel update management, and capability restriction directly govern the mitigations described in this

note. Organizations using the AICM as their AI security control framework should map Dirty Frag exposure to those infrastructure and runtime security controls and document their remediation status accordingly.

CSA's "Best Practices for Implementing a Secure Application Container Architecture" provides complementary operational guidance for the container-specific mitigations most relevant to AI workloads [19]. This publication addresses container host hardening, runtime security configurations, and privilege minimization across the developer, operator, and architect roles — all directly applicable to the seccomp and capability restriction improvements this note recommends.

The Zero Trust principles articulated across CSA's Zero Trust guidance apply directly to the architectural problem that Dirty Frag surfaces: a shared kernel creates an implicit trust relationship between co-resident containers that violates Zero Trust's mandate for explicit, verified authorization at every access boundary. The long-term architectural response to this vulnerability class — node-level tenant isolation, enforced pod security standards, and elimination of unnecessary privilege grants — aligns with Zero Trust's core principle of minimizing implicit trust and reducing blast radius.

References

[1] Bleeping Computer. "New Linux 'Dirty Frag' Zero-Day Gives Root on All Major Distros." Bleeping Computer, May 2026.

[2] Heise Online. "'Dirty Frag': Linux Flaws Grant Root Access." Heise Online, May 2026.

[3] V4bel. "dirtyfrag – Technical Write-up." GitHub, May 2026.

[4] Wiz Security Research. "Dirty Frag (CVE-2026-43284): Linux Kernel Local Privilege Escalation via ESP and RxRPC." Wiz Blog, May 2026.

[5] The Hacker News. "Linux Kernel Dirty Frag LPE Exploit Enables Root Access Across Major Distributions." The Hacker News, May 2026.

[6] Security Boulevard. "Dirty Frag (CVE-2026-43284, CVE-2026-43500): Frequently Asked Questions About This Linux Kernel Privilege Escalation Vulnerability Chain." Security Boulevard, May 2026.

[7] Sysdig. "Dirty Frag (CVE-2026-43284 and CVE-2026-43500): Detecting Unpatched Local Privilege Escalation via Linux Kernel ESP and RxRPC." Sysdig Blog, May 2026.

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

[9] Help Net Security. "Dirty Frag: Unpatched Linux Vulnerability Delivers Root Access." Help Net Security, May 2026.

[10] AlmaLinux. "Dirty Frag (CVE-2026-43284, CVE-2026-43500) Patches Released." AlmaLinux Blog, May 2026.

[11] Ubuntu. "Dirty Frag Linux Kernel Local Privilege Escalation Vulnerability Mitigations." Ubuntu Blog, May 2026.

[12] Canonical. "Dirty Frag Linux Kernel Local Privilege Escalation Vulnerability Mitigations." Canonical Blog, May 2026.

[13] CloudLinux. "Dirty Frag: Mitigation and Kernel Update on CloudLinux." CloudLinux Blog, May 2026.

[14] Red Hat. "RHSB-2026-003: Networking Subsystem Privilege Escalation – Linux Kernel (Dirty Frag) – CVE-2026-43284." Red Hat Customer Portal, May 2026.

[15] Microsoft Threat Intelligence. "Active Attack: Dirty Frag Linux Vulnerability Expands Post-Compromise Risk." Microsoft Security Blog, May 2026.

[16] Juliet. "Dirty Frag in Kubernetes: EKS and GKE Exposed With Unset Seccomp." Juliet Blog, May 2026.

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

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

[19] Cloud Security Alliance. "Best Practices for Implementing a Secure Application Container Architecture." Cloud Security Alliance, July 2019.