WHITE PAPER

CYTEX™
AI-Powered Resilience Command Center

Fragnesia

A Race-Condition-Free Linux LPE Born From a Security Fix

hello@cytex.io | https://cytex.io

Fragnesia

Technical analysis of the latest entry in the Dirty Frag vulnerability class, with detection signals, mitigation guidance, and defensive implications for containerized and multi-tenant Linux environments.

On May 13, 2026, the V12 Security team publicly disclosed Fragnesia, a local privilege escalation vulnerability in the Linux kernel's XFRM ESP-in-TCP subsystem, tracked as **CVE-2026-46300**. A working proof-of-concept was released the same day. The vulnerability affects every Linux kernel built before May 13, 2026 that was previously affected by the Dirty Frag class.

Fragnesia matters for three reasons that distinguish it from a typical kernel CVE disclosure:

One. It requires no race condition. Unlike most kernel-memory corruption primitives, the exploit is deterministic, a property that lifts exploitation reliability from "works sometimes" to "works consistently across runs."

Two. It emerged as an unintended consequence of patches shipped to fix the *original* Dirty Frag vulnerabilities (CVE-2026-43284 and CVE-2026-43500). The fix introduced a new exploitable path in the same subsystem. Organizations that applied only the upstream Dirty Frag patches and skipped the module blacklist mitigation are not protected against Fragnesia.

Three. A public proof-of-concept is already on GitHub. A threat actor identified as **berz0k** is concurrently advertising a Linux LPE zero-day on cybercrime forums for $170,000. While the two are not confirmed to be the same exploit, the market signal, paid demand for universal Linux LPE in mid-May 2026, coincides with the timing of free public PoCs for Dirty Frag, Copy Fail, and now Fragnesia.

No in-the-wild exploitation has been observed at the time of this writing. The patch is available upstream and from major distributions. The window between disclosure and weaponization in equivalent recent kernel LPE classes has been measured in hours.

The Vulnerability

Attribute	Detail
CVE	CVE-2026-46300
Name	Fragnesia
Vulnerability class	Local privilege escalation (LPE)
Affected subsystem	Linux kernel XFRM ESP-in-TCP (skbuff path)

Attribute	Detail
Root-cause function	skb_try_coalesce() — fails to propagate SKBFL_SHARED_FRAG
Required configuration	CONFIG_INET_ESPINTCP enabled in the kernel build
Required capabilities	None — exploitable from an unprivileged user namespace
Race condition required	**No**
CVSS score	7.8 (High)
Discovered by	William Bowling, Zellic / V12 Security
Disclosure date	May 13, 2026
Public PoC	Available on GitHub the same day
In-the-wild exploitation	None observed as of this writing
Related CVEs	CVE-2026-43284 (Dirty Frag — xfrm-ESP page-cache write), CVE-2026-43500 (Dirty Frag — RxRPC page-cache write), CVE-2026-31431 (Copy Fail — already in CISA KEV)

Vulnerability Class Family

Fragnesia is the third entry in roughly three weeks in the same general bug class:

- **Copy Fail** (CVE-2026-31431) — added to CISA KEV on May 1, 2026
- **Dirty Frag** (CVE-2026-43284 + CVE-2026-43500, chained) — disclosed days before Fragnesia
- **Fragnesia** (CVE-2026-46300) — disclosed May 13, 2026

All three achieve the same outcome through different code paths in the kernel's networking and page-cache layers: arbitrary writes into the in-memory page cache of read-only files, including privileged setuid binaries.

How Fragnesia Works

The Root-Cause Defect

The flaw lives in skb_try_coalesce(), a function in the kernel's socket buffer (skb) handling code. When the kernel coalesces socket buffer fragments to merge multiple skbs into a single buffer, the function fails to propagate the SKBFL_SHARED_FRAG flag onto the merged buffer when the source fragments are marked as shared with other subsystems, including the page cache.

Without that flag, the kernel treats the merged buffer's pages as private and safe to write in place. In reality, those pages are still page-cache-backed copies of read-only files on disk. The kernel's ownership semantics break down at this boundary: the buffer thinks it owns the pages; the page cache still thinks the pages are immutable file-backed memory.

The candidate fix carries a Fixes: tag pointing to a 2013 commit. The defect has been latent in the kernel for over a decade.

The Exploitation Sequence

The published proof-of-concept performs the following steps, all from an unprivileged user:

Step 1 — Acquire a sandboxed CAP_NET_ADMIN. The exploit calls unshare(CLONE_NEWUSER | CLONE_NEWNET) to enter a new user namespace and a new network namespace. Inside the new user namespace, the attacker holds CAP_NET_ADMIN within the sandbox, sufficient to configure XFRM/IPsec state for the namespace's sockets without requiring any privileges on the host.

Step 2 — Install an ESP-in-TCP security association. Via NETLINK_XFRM, the exploit installs a transport-mode ESP-in-TCP security association using AES-128-GCM with a known, attacker-controlled key. The known key is what makes the byte-write primitive deterministic, the attacker controls both the ciphertext input and the IV, and therefore controls the plaintext output produced by in-place decryption.

Step 3 — Splice file-backed pages into the TCP receive queue. Using splice(), the attacker moves pages backed by a read-only target file (for example /usr/bin/su) into the receive queue of a TCP socket *before* switching the socket into espintcp mode via the kernel's ULP (Upper Layer Protocol) framework.

Step 4 — Trigger coalescing. The exploit drives the receive path such that skb_try_coalesce() merges fragments referencing those file-cache-backed pages. The SKBFL_SHARED_FRAG flag fails to propagate. The kernel now considers the shared cached pages to be writable buffers.

Step 5 — Decrypt in place over the cached file pages. The kernel's ESP-in-TCP processing path decrypts the attacker-supplied ciphertext using the attacker-controlled key, and writes the decryption output directly into the cached pages of the target file. Because the attacker controls the key and the IV, the decryption transform becomes a controlled one-byte-at-a-time write primitive against the in-memory copy of /usr/bin/su.

Step 6 — Write a malicious ELF stub into the cached binary. Iterating the byte-write primitive, the exploit writes a small ELF stub into the cached copy of /usr/bin/su. The on-disk binary remains unmodified. Standard file integrity monitoring tools that hash the disk inode see no change.

Step 7 — Invoke su to spawn a root shell. When the user next invokes su, the kernel loads the cached copy from the page cache, which now contains attacker-controlled code. The stub executes as root.

What Makes This Class Particularly Dangerous

The on-disk file is never modified. Hash-based file integrity monitors, package verification (rpm -V, dpkg --verify), and most endpoint controls that compare files to known-good baselines will not detect Fragnesia exploitation. The compromise lives entirely in volatile memory until the page is evicted.

The exploit does not require a race. Race-condition LPEs typically require many attempts and have variable reliability. Fragnesia's deterministic byte-write primitive means an attacker can rely on it succeeding on a single attempt under normal system load.

The target is not limited to /usr/bin/su. Any file the unprivileged user can *read*, meaning any world-readable file with elevated meaning to the system, is a viable target. /etc/passwd, /etc/shadow (where readable), /etc/sudoers, and any setuid binary become candidates.

Container escape becomes plausible. An unprivileged container that allows user namespace creation (a common default for rootless Docker and Podman) gives the exploit everything it needs. The exploit does not require privileged containers, capabilities, or host network access. Multi-tenant Kubernetes worker nodes, CI runners, and shared container hosts are first-class targets.

Affected Systems

Kernel scope. Every Linux kernel version affected by Dirty Frag, built before **May 13, 2026**, is also affected by Fragnesia unless explicitly patched against CVE-2026-46300.

Distribution scope. Confirmed-affected major distributions include:

- Red Hat Enterprise Linux (RHSB-2026-003 covers CVE-2026-43284, CVE-2026-43500, and CVE-2026-46300)
- Ubuntu
- Debian
- SUSE Linux Enterprise
- AlmaLinux
- Amazon Linux
- CloudLinux

Configuration scope. The public PoC requires CONFIG_INET_ESPINTCP to be enabled in the kernel build to reach the bug through that path. Kernels built without CONFIG_INET_ESPINTCP are not exploitable through the public PoC, but the underlying skbuff defect may be reachable through other paths and should still be patched.

Patch divergence from Dirty Frag. Organizations that applied only the upstream kernel patches for the original Dirty Frag CVEs (CVE-2026-43284 and CVE-2026-43500) are **not protected** against Fragnesia. A separate patch is required. Organizations that applied the Dirty Frag *module blacklist mitigation* (disabling esp4, esp6, and rxrpc) are protected against both Dirty Frag and Fragnesia because the same mitigation removes the attack surface for both.

Detection Signals: Operational IOCs and Behavioral Indicators

Fragnesia is a local exploitation primitive. There are no network IOCs, no malware family signatures, no command-and-control domains, and no file hashes specific to the vulnerability itself, exploitation is performed by the legitimate kernel handling unusual but valid syscalls. The detection signal is **behavioral**, focused on the syscall and namespace patterns the public PoC and similar exploits will produce.

The following behaviors, observed on a Linux host, warrant investigation:

Signal	Why It Matters
`unshare(CLONE_NEWUSER CLONE_NEWNET)` calls from unprivileged users in workloads that do not legitimately use user namespaces	
NETLINK_XFRM activity from workloads with no IPsec or VPN role	The exploit installs an XFRM security association via netlink. Hosts that do not run IPsec should not see XFRM netlink traffic from user-space workloads.
Heavy AF_ALG crypto interface usage in tight loops	The exploit drives the kernel's crypto path repeatedly to iterate the byte-write primitive. Sustained tight-loop AF_ALG activity is anomalous on most workloads.
splice() from a regular file into a TCP socket, followed by ULP mode change on the same socket	This is the exact precondition the exploit depends on. Almost no legitimate workload combines these in this order.
Modifications to the cached copy of /usr/bin/su, /etc/passwd, /etc/shadow, /etc/sudoers not reflected on disk	The defining symptom. A diff between the in-memory and on-disk views of these files is itself an exploitation indicator.

Signal	Why It Matters
Unexpected root shells spawned from non-root parent processes immediately after invoking su	The post-exploitation tell. Worth correlating against process ancestry telemetry.

eBPF-based detection layers (Falco, Tetragon, Tracee) are the right place to instrument these signals. Standard EDR product coverage for this class is uneven; the syscall sequence is unusual enough that custom rules are typically required.

No file-integrity-monitoring (FIM) signal. This is worth restating. Tools that compare on-disk file contents to known-good baselines will produce no alert during Fragnesia exploitation. The compromise is entirely page-cache resident.

Defensive Actions

Patch

Apply the kernel update for CVE-2026-46300 from your distribution as soon as available. Vendor advisories to track:

- Red Hat: **RHSB-2026-003** (covers CVE-2026-43284, CVE-2026-43500, CVE-2026-46300)
- SUSE: VUL-0 advisory referencing CVE-2026-46300
- Ubuntu: USN search on CVE-2026-46300
- Debian, AlmaLinux, Amazon Linux, CloudLinux: track each vendor's security feed

For environments where reboot windows are constrained, rebootless kernel-patching solutions (e.g., KernelCare) have published live patches for CloudLinux 9 ELS and AlmaLinux 9 ESU as of May 14, 2026, 07:00 UTC.

If Patching Is Delayed:

The Dirty Frag mitigation also addresses Fragnesia. Unload and blacklist the vulnerable kernel modules:

```
# Unload the modules

modprobe -r esp4 esp6 rxrpc

# Persist via blacklist

cat > /etc/modprobe.d/dirty-frag-mitigation.conf <<EOF

blacklist esp4

blacklist esp6

blacklist rxrpc

install esp4 /bin/true
```

install esp6 /bin/true

install rxrpc /bin/true

EOF

Operational caveat. Disabling esp4 and esp6 breaks workloads that rely on IPsec ESP. Disabling rxrpc breaks AFS/RxRPC workloads. Apply only on systems where these dependencies are confirmed absent or operationally acceptable to disable.

If a System May Already Have Been Targeted

The exploit corrupts memory but not disk. To clean a potentially exploited host *after* mitigation has been applied, drop the page cache so corrupted cached pages are evicted and reloaded fresh from disk:

sync

echo 3 > /proc/sys/vm/drop_caches

Important. Dropping the cache cleans the memory artifact only. If an attacker successfully escalated to root and installed a persistent backdoor, cache eviction does not remove the backdoor. Any host confirmed to have been exploited should be treated as a full compromise, investigated, rebuilt, and credentials rotated.

Defense-in-Depth for Container and Multi-Tenant Environments

For Kubernetes worker nodes, shared CI runners, and multi-tenant container platforms, additional hardening reduces the blast radius:

- **Disable unprivileged user namespace creation** where operationally feasible: sysctl kernel.unprivileged_userns_clone=0 (Debian/Ubuntu) or equivalent AppArmor profile restrictions. Ubuntu hosts running AppArmor namespace restrictions have partial protection against the public PoC.
- **Seccomp profiles** that disallow AF_ALG socket creation for workloads that do not require kernel crypto interfaces. The exploit's tight-loop AF_ALG usage will be blocked at the syscall layer.
- **Restrict unshare, add_key, and splice** for unprivileged users via seccomp where workload requirements allow.
- **Audit container security profiles** to ensure rootless container runtimes are not unintentionally granting user namespace creation in security boundaries that assume otherwise.

Monitoring

- Add detection rules for the behavioral indicators to your eBPF runtime security tooling and SIEM.

- Increase monitoring for unexpected setuid binary executions whose parent process lineage is unusual.
- Watch for filesystem-vs-page-cache divergence on critical setuid binaries and /etc/ files where instrumentation supports it.

The Threat Actor: berz0k

A threat actor operating under the handle **berz0k** is advertising a zero-day Linux LPE for $170,000 on cybercrime forums, claiming functionality across multiple major Linux distributions. There is no public confirmation that berz0k's offering is Fragnesia, Dirty Frag, Copy Fail, or a separate, unreleased flaw. The relevant signal for defenders is timing, paid demand for universal Linux LPE is present in the same window in which three distinct, freely available LPE PoCs (Copy Fail, Dirty Frag, Fragnesia) have been published.

The market for Linux LPE is liquid, the buyer base is willing to pay six figures, and the supply side is producing usable exploits faster than enterprise patch cycles can absorb them. Defenders should not wait for confirmed in-the-wild exploitation reports before treating these vulnerabilities as imminent.

Why This Class Is Not Finished

Fragnesia is the third Linux kernel LPE in this general code-path neighborhood in roughly three weeks. The pattern matters more than any individual CVE:

- The fix for the original Dirty Frag introduced Fragnesia. **Security fixes for complex memory-handling bugs are themselves becoming targets for rapid bypass research.**
- The defect at the root of Fragnesia traces to a 2013 commit. **The bug class has been latent in the kernel for over a decade**, and the discovery cadence is accelerating now that researchers, and AI-assisted tooling, are looking at it systematically.
- The XFRM, splice, skb-coalescing, and page-cache subsystems are all *performance-oriented* paths. Performance optimizations relax ownership semantics; relaxed ownership semantics produce exploitable invariant violations. **The same engineering pressure that produces these subsystems produces the bugs.**

Defenders should plan for additional disclosures in this class over the coming months and treat the broader skbuff/ESP/page-cache surface as a category-level concern, not an incident-level one.

Fragnesia is a high-severity Linux kernel local privilege escalation that turns any foothold on a vulnerable system into root. The exploitation primitive is deterministic, the proof-of-concept is public, the on-disk file remains unmodified throughout exploitation, and the attack works through an unprivileged container's user namespace.

Three uncomfortable truths for defenders to internalize:

First, file-integrity monitoring as traditionally configured does not see this class of attack. Defensive telemetry has to instrument page-cache and syscall behavior, not just disk state.

Second, the patch for one vulnerability in this class produced the next vulnerability in this class. Patch hygiene is necessary but not sufficient; the architectural attack surface, namespace-accessible kernel subsystems with performance-relaxed ownership semantics, remains intact.

Third, the disclosure-to-exploitation window for Linux LPE in this class has been measured in hours during 2026. The defensive timeline that calibrates to weekly patch cycles is structurally mismatched to the threat velocity. Patch immediately, mitigate where you cannot patch, and treat the broader bug class as an ongoing operational concern rather than a closed incident.

Sources:

- SOC Prime: *CVE-2026-46300: Fragnesia Linux Kernel Flaw*, May 2026.
- Red Hat Security Bulletin RHSB-2026-003 (Networking subsystem Privilege Escalation — Linux Kernel — Dirty Frag family).
- Original disclosure: William Bowling, Zellic / V12 Security, public posting and PoC on GitHub, May 13, 2026.