FragLab v0.1 CopyFail LXC Validation

Testing CVE-2026-31431 in Privileged vs. Unprivileged LXC

Author: Saar Yachin
Date: 17 May 2026
Environment: Proxmox VE / Ubuntu LXC homelab
Purpose: Controlled exploitability validation

Executive Summary

FragLab v0.1 tested CopyFail (`CVE-2026-31431`) inside two Ubuntu 24.04 LXC containers on the same Proxmox host: one privileged and one unprivileged.

The main finding was that LXC privilege mode materially affected practical exploitability. The CopyFail PoC failed in the unprivileged LXC before reaching the escalation stage, but succeeded in the privileged LXC and produced a root shell inside the container.

After the Proxmox host was updated and rebooted from kernel `6.17.13-2-pve` to `7.0.2-4-pve`, the same PoC still succeeded inside the privileged LXC.

No Proxmox host escape was demonstrated or claimed. The successful result is classified as **container-local privilege escalation**.

CopyFail Background

CopyFail, tracked as `CVE-2026-31431`, is a Linux kernel local privilege escalation vulnerability publicly disclosed on April 29, 2026. The vulnerability affects the Linux kernel `algif_aead` / `AF_ALG` crypto interface and can allow an unprivileged local user to corrupt page-cache data associated with setuid binaries, leading to root privilege escalation. Public reporting and the PoC repository attribute the discovery to Theori's Xint Code research.

This report does not analyze the vulnerability internals in depth. It focuses on practical validation inside privileged and unprivileged LXC environments. Public sources were used only for vulnerability background; the experiment below is limited to practical lab validation.

Scope

This experiment tested one public CopyFail PoC against two Ubuntu 24.04 LXC containers on one Proxmox host. It did not include exploit development, host escape research, detection engineering, mitigation testing, or third-party systems.

Lab Environment

Field	Value
Proxmox host	`dcitadel`
Host OS before update	Debian GNU/Linux 13.4 “trixie”
Host OS after update	Debian GNU/Linux 13.5 “trixie”
Proxmox VE	`9.1.0`
Initial running kernel	`6.17.13-2-pve`
Post-update running kernel	`7.0.2-4-pve`

LXC containers share the Proxmox host kernel. Therefore, the Proxmox running kernel was treated as the vulnerability-relevant layer. The Ubuntu version inside each container was treated as userland context.

Host baseline **before update**:

```
saar@dcitadel:~$ uname -a
Linux dcitadel 6.17.13-2-pve #1 SMP PREEMPT_DYNAMIC PMX 6.17.13-2 (2026-
03-13T08:06Z) x86_64 GNU/Linux
saar@dcitadel:~$ sudo pveversion -v
proxmox-ve: 9.1.0 (running kernel: 6.17.13-2-pve)
pve-manager: 9.1.9
lxc-pve: 6.0.5-4
pve-container: 6.1.5
```

Proxmox host baseline **after update and reboot**:

```
saar@dcitadel:~$ uname -a
Linux dcitadel 7.0.2-4-pve #1 SMP PREEMPT_DYNAMIC PMX 7.0.2-4 (2026-05-
15T07:32Z) x86_64 GNU/Linux
saar@dcitadel:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 13 (trixie)"
VERSION_ID="13"
VERSION="13 (trixie)"
DEBIAN_VERSION_FULL=13.5
```

Target Containers

VMID	Hostname	IP Address	LXC Type	OS	Resources
`482`	`fraglab-unpriv`	`172.16.66.104`	Unprivileged LXC	Ubuntu 24.04 LTS	1 CPU, 1 GB RAM, 4 GB disk
`483`	`fraglab-priv`	`172.16.66.105`	Privileged LXC	Ubuntu 24.04 LTS	1 CPU, 1 GB RAM, 4 GB disk

`fraglab-unpriv` was confirmed with `unprivileged: 1` in Proxmox; `fraglab-priv` had no `unprivileged: 1` flag.

Both containers used the same non-sudo test user: `unprivuser`.

PoC Source and Integrity

The PoC used in this experiment was taken from the public GitHub repository `painoob/Copy-Fail-Exploit-CVE-2026-31431`. The python script used in testing was verified with SHA256: `818341db590663bd49c3517966509856049449d124c620fd3c2fbf887cdb4ce3`.

The same code was used across both LXC targets and across the pre-/post-update tests.

Hypothesis

Because both containers share the same Proxmox host kernel, the kernel version was relevant to both targets. However, the practical result was expected to differ because privileged and unprivileged LXC containers expose different security contexts.

Target	Hypothesis
Unprivileged LXC	The PoC may fail or be contained due to user namespace restrictions and reduced access to kernel interfaces.
Privileged LXC	The PoC has a higher chance of succeeding due to weaker isolation.
Host escape	Not expected and not tested. This was treated as local privilege escalation validation, not a container breakout test.

Results Matrix

Test	Target	LXC Type	Kernel	Vulnerability Layer	Result	Classification
1	`fraglab-unpriv`	Unprivileged	`6.17.13-2-pve`	Shared host kernel	Failed with `Errno 97` during socket creation	No escalation
2	`fraglab-priv`	Privileged	`6.17.13-2-pve`	Shared host kernel	Root shell obtained inside container	Container-local LPE success

Test	Target	LXC Type	Kernel	Vulnerability Layer	Result	Classification
3	`fraglab-priv`	Privileged	`7.0.2-4-pve`	Shared host kernel	Root shell obtained inside container	Container-local LPE success

Test 1: Unprivileged LXC

Target:

```
Hostname: fraglab-unpriv
IP: 172.16.66.104
Type: Unprivileged LXC
Kernel: 6.17.13-2-pve
User: unprivuser
```

The PoC failed during socket creation:

```
unprivuser@fraglab-unpriv:~$ ./copyfail.py
Traceback (most recent call last):  ...OSError: [Errno 97] Address family
not supported by protocol
```

Relevant failing operation:

```
socket(38, 5, 0)
```

Classification: **No escalation**

Interpretation:

The failure occurred before the PoC reached the privilege escalation stage. CopyFail relies on the Linux kernel crypto API exposed through `AF_ALG`. In this test, the unprivileged LXC did not expose the required socket path to the unprivileged user context. This prevented practical exploitation in this configuration.

Test 2: Privileged LXC Before Host Update

Target:

```
Hostname: fraglab-priv
IP: 172.16.66.105
Type: Privileged LXC
```

```
Kernel: 6.17.13-2-pve
User: unprivuser
```

The PoC succeeded and produced a root shell inside the privileged container.

```
unprivuser@fraglab-priv:~$ ./copyfail.py
rootbash-5.2# whoami
root
rootbash-5.2# id
uid=0(root) gid=1000(unprivuser) groups=1000(unprivuser),100(users)
```

Classification: **Container-local privilege escalation succeeded**

Interpretation:

The PoC escalated `unprivuser` to UID 0 inside the privileged LXC. This demonstrates local privilege escalation within the container. It does not demonstrate Proxmox host escape.

Note: in this first successful run, the shell became UID 0 while retaining the original user's GID. This was recorded but not investigated further in v0.1.

Test 3: Privileged LXC After Host Update and Reboot

After the initial tests, the Proxmox host was updated and rebooted.

Post-update state:

```
Host kernel: 7.0.2-4-pve
Host OS: Debian GNU/Linux 13.5 "trixie"
```

Inside the privileged LXC, the container reported the shared updated host kernel:

```
unprivuser@fraglab-priv:~$ uname -a
Linux fraglab-priv 7.0.2-4-pve #1 SMP PREEMPT_DYNAMIC PMX 7.0.2-4 (2026-
05-15T07:32Z) x86_64 x86_64 x86_64 GNU/Linux
```

The same PoC was rerun as `unprivuser`.

```
unprivuser@fraglab-priv:~$ ./copyfail.py
rootbash-5.2# whoami
root
rootbash-5.2# id
uid=0(root) gid=0(root) groups=0(root)
```

Classification: **Container-local privilege escalation succeeded**

Interpretation:

CopyFail still succeeded inside the privileged LXC after the Proxmox host was updated and rebooted into kernel `7.0.2-4-pve`. This does not prove host compromise. It shows that the privileged LXC remained practically exploitable by this PoC in this lab configuration.

In this second successful run, the resulting shell had both UID 0 and GID 0. The difference from the first run was recorded but not investigated further.

Key Findings

1. **LXC privilege mode affected practical exploitability.**
   The same PoC failed in the unprivileged LXC but succeeded in the privileged LXC.
2. **The successful result was container-local.**
   CopyFail produced UID 0 inside the privileged LXC. No Proxmox host escape was demonstrated or claimed.
3. **The privileged LXC remained exploitable after host update/reboot.**
   After the host moved from kernel `6.17.13-2-pve` to `7.0.2-4-pve` — a kernel built on 15 May 2026 and tested on 17 May 2026 — the same PoC still produced root inside the privileged LXC. This suggests a practical patch-to-remediation gap in the tested configuration.
4. **Kernel/userland distinction was critical.**
   The Ubuntu 24.04 containers shared the Proxmox host kernel. `uname -a` inside the container reflected the host kernel, while `/etc/os-release` reflected container userland.
5. **`/usr/bin/su` was container-local.**
   The `su` binary observed inside the container belonged to the container filesystem. This does not imply modification of the Proxmox host's `/usr/bin/su`.

Security Recommendation

Avoid privileged LXC containers for untrusted workloads.

In this test, unprivileged LXC isolation prevented the PoC from reaching the vulnerable path, while the privileged LXC allowed container-local root compromise both before and after the host update/reboot. Root inside a container is not automatically host root, but privileged containers significantly reduce the safety margin.

Limitations

This was a narrow v0.1 experiment. Results should not be generalized beyond the tested setup.

Limitations:

- only one PoC was tested

- only Ubuntu 24.04 LXC was tested
- only one Proxmox host was tested
- no host escape was attempted
- no detection or mitigation analysis was performed

Conclusion

FragLab v0.1 showed that LXC privilege mode was the decisive observed variable. The same CopyFail PoC failed in the unprivileged LXC but produced container-local root in the privileged LXC, both before and after the Proxmox host update/reboot. The result reinforces the importance of avoiding privileged containers for untrusted workloads and of distinguishing carefully between container root and host compromise.

Further Research

Potential next research steps for FragLab v1.0:

- test additional kernel PoCs such as DirtyFrag and Fragnesia
- repeat the unprivileged LXC test after the host update
- add a full VM baseline
- test additional Linux distributions
- compare Ubuntu, Debian, Rocky, Fedora, and Alpine-style environments