CVE-2026-31431 (Copy Fail) Vulnerability FAQ (Part 1) — Basic Information, Scope of Impact, and Handling of the Vulnerability

The original report is in Chinese, and this version is an AI-translated edition.

Disclaimer: This article is based on a FAQ list compiled by Antiy CERT engineers, completed collaboratively by multiple AI agents, with some content manually revised and rewritten. Although we have manually verified the information, due to the large amount of information, the AI-generated content may contain factual inaccuracies or timeliness errors. Technical verification cannot be completed in a short time, so readers are advised to cross-verify. If you find any errors or have any questions, please leave a message directly. We will continue to revise the website version of the FAQ after verification.

Editor's Note

On April 29, 2026, South Korean security company Theori and its AI security research platform Xint publicly disclosed a Linux kernel native privilege escalation vulnerability, CVE-2026-31431, codenamed "Copy Fail". Antiy CERT is closely monitoring and responding to this vulnerability.

This vulnerability exists in the Linux kernel cryptosystem, affecting almost all major Linux distributions since 2017. Attackers only need local ordinary user privileges and can reliably gain root privileges under uncontested conditions using a Python script of less than 800 bytes. The vulnerability achieves execution hijacking by polluting the setuid program in the page cache and has the potential to escape into containers under certain environment configurations (affecting file caches in the host machine's memory, rather than the disk files themselves).

Antiy CERT has compiled this FAQ to provide accurate and actionable threat awareness and response guidance for different audiences.

Explanation of the article's structure:

■ **Part 1 (This Article)**: Primarily aimed at the general public, network administrators, network users, industry regulatory authorities, and corporate IT decision-makers. It focuses on basic vulnerability awareness, impact assessment, asset identification, remediation and mitigation measures, and threat posture evaluation. The writing style strives for clarity and accuracy and can be directly used as reference material for internal communications and management reports.

■ **Part 2 (Preview)**: Targeting security researchers, analysis engineers, kernel developers, and advanced blue team personnel. It focuses on the underlying technical mechanisms of vulnerabilities, exploit chain disassembly, comparative analysis with historical vulnerabilities, and strategic reflections on Linux kernel security auditing mechanisms, offering a high level of technical depth.

The two articles complement each other and cite the same sources, both based on cross-validation of publicly disclosed information, the oss-security mailing list, upstream kernel patches, and community testing feedback.

1. Basic understanding of vulnerabilities

Q1: What is the CVE-2026-31431 (Copy Fail) vulnerability, and what is the technical meaning behind its name?

A: CVE-2026-31431 is a high-risk local privilege escalation vulnerability in the Linux kernel, codenamed "Copy Fail".

In layman's terms: To improve performance, the Linux kernel employs an "in-place read/write" optimization when handling certain encryption operations. This means that data is read in and modified directly at its original location, rather than being copied to a secure memory area first. The problem arises when this data comes from the cache of ordinary files (page cache). The kernel mistakenly treats "read-only" file cache pages as "writable" buffers. Attackers can exploit this flaw to write malicious code into the memory cache of any readable file on the system (including privilege escalators like `/usr/bin/su`), and then execute that program to gain root privileges.

The name "Copy Fail" comes from this: the "copy" operation that should have been performed failed, causing the data to be modified in places where it shouldn't have been.

Q2: How is the score and severity level of this vulnerability defined under the CVSS 3.1 framework?

A: The CVSS 3.1 base score for this vulnerability is **7.8, which is classified as "High"**. **However, this vulnerability has a very wide coverage, can be penetrated in many versions, and has a very high success rate when used in combination with other vulnerabilities. The risk is clearly underestimated by CVSS.**

Evaluation dimensions	Rating	Illustrate
Attack vector	Local	Attackers need to have local execute privileges on the target system.
Attack complexity	Low	It has a low barrier to entry and requires no complicated operation.
Required permissions	Low	Only requires regular user privileges, no root access required.
User interaction	None	No need to trick users into clicking or interacting
Scope of influence	Constant	Affects the same system under attack
Confidentiality impact	High	Can read any sensitive data
Integrity impact	High	Tamperable critical system files
Availability impact	High	This can lead to system crashes or service interruptions.

Although the CVSS scoring framework classifies it as "high risk", in scenarios such as **multi-tenant cloud environments, container clusters, and CI/CD pipelines**, its actual risk is close to "Critical" - because a single ordinary user's privilege escalation can compromise the security isolation of the entire host machine or cluster.

Q3: Is this vulnerability a local privilege escalation (LPE) or remote code execution (RCE) vulnerability?

A: **Local privilege escalation (LPE)**.

Attackers cannot directly trigger this vulnerability remotely over a network. They must first gain local access to the target system, for example:

■ Regular user accounts logged in via SSH

■ Local command execution privileges obtained through web application vulnerabilities (Web RCE)

■Non-privileged users within the container

■ Untrusted code executed in the CI/CD pipeline

Once a local foothold is gained, an attacker can run a tiny Python script (only 732 bytes) to escalate privileges from a regular user to root. Therefore, for systems with existing web vulnerabilities or container intrusions, Copy Fail can quickly escalate a "partial intrusion" to "full control".

Q4: In which version cycle was the vulnerability code introduced, and why was it able to remain dormant in the kernel for about nine years?

A:

Introduction Date: 2017, Linux kernel submission `72548b093ee3` ("crypto: algif_aead - add in-place AEAD support"). **Versions Implemented**: Linux 4.14 and later. **Impact Window**: 2017 to April 2026, approximately nine years.

How was he able to remain hidden for nine years?

1. Involves coupling of multiple subsystems: The vulnerability requires a "precise" combination of four independent subsystems to be triggered: `the AF_ALG` encryption interface, `splice()` zero-copy transmission, the `algif_aead` kernel module, and the page cache. Such cross-system interaction flaws are difficult to detect through code auditing of a single subsystem.

2. The illusion of "reasonableness" in optimizing code: The in-place optimization introduced in 2017 is a reasonable improvement from a performance perspective. However,

the problem is triggered by the need to pass in the file cache page in conjunction with the splice() system call. Each of them seems correct on its own, but the combination has fatal consequences.

3. Test coverage blind spot: Kernel regression testing failed to cover the edge case of "splice() passing a page cache page to the encryption interface for decryption".

4. AI-assisted discovery: The vulnerability was ultimately discovered from the crypto/subsystem by Xint's AI-assisted code auditing tool in about 1 hour, demonstrating that traditional manual auditing is difficult to cover such complex coupling defects.

Q5: When was this vulnerability first discovered, and when was it officially disclosed to the public?

A:

Time	Event
March 23, 2026	Theori/Xint researcher Taeyang Lee reported a vulnerability to the Linux kernel security team.
March 24, 2026	The kernel security team has made a preliminary confirmation.
March 25, 2026	The patch has been submitted and is now under code review.
April 1, 2026	The patch has been merged into the Linux mainline kernel (commit a664bf3d603d).
April 22, 2026	CVE-2026-31431 officially assigned
April 29, 2026	The PoC is publicly disclosed at https://copy.fail/
April 30, 2026	Major distributions begin pushing out security patches

The time from report to public disclosure was approximately 37 days, which is consistent with the 90-day disclosure convention for Linux kernel security vulnerabilities.

2. Scope of impact and assets

Q6: Which major Linux kernel versions and LTS branches have been confirmed to be affected by this vulnerability?

A:

Affected version range: Linux 4.14 (the defect was introduced in 2017) to all versions before the patch was merged.

Kernel branch	State	Illustrate
Linux 4.14 ~ 4.19 LTS	Affected	Some have already reached end-of-life (EOL) and may not have official patches.
Linux 5.4 ~ 5.15 LTS	Affected	The mainstream LTS branch and distribution are currently receiving backport patches.
Linux 6.1 ~ 6.12 LTS	Affected	Current mainstream production environment version
Linux 6.18	Affected	Security version ≥ 6.18.22
Linux 6.19	Affected	Security version ≥ 6.19.12
Linux 7.0 mainline	Affected (before repair)	Security after including commit a664bf3d603d
Linux 4.9 and earlier	Unaffected	Defects have not yet been introduced

***Important Note**: Distributions typically **backport patches to the existing kernel version**, not necessarily by upgrading the kernel major version number. For example, Ubuntu might include a patch in kernel 6.8.0-55 instead of upgrading to 6.18.22. Therefore, **security cannot be determined solely by the major version number obtained using `uname -r`**; you must verify the distribution's official security advisories and the specific package version.*

Q7: How are mainstream commercial distributions and domestic operating systems affected?

A:

International release (verified)

Release version	Version	State
Ubuntu	16.04 LTS ~ 26.04 LT	Affected

	S	
Red Hat Enterprise Linux	8/9 series	Affected
SUSE Linux Enterprise	15 SP5/16	Affected
Debian	bullseye/bookworm/trixie	Affected
Amazon Linux	2/2023	Affected
Fedora	40/41/42	Affected
Arch Linux	Rolling release	Affected
Rocky / Alma / Oracle Linux	8/9	Affected

Domestic operating system

Release version	Kernel basics	State	Suggestion
Kylin operating system	Based on RHEL/Ubuntu	Theoretically affected	Pay attention to official security announcements and confirm your kernel version.
Tongxin UOS	Based on Debian	Theoretically affected	Pay attention to official security announcements
openEuler	Based on Linux kernel	Theoretically affected	Security advisory has been released, and patches are being pushed out.
Anolis OS	Based on RHEL	Theoretically affected	Follow the upstream patch
Deepin OS	Based on Debian	Theoretically affected	Follow Debian patches
NFS	Based on Linux	Theoretically affected	Contact the manufacturer to obtain the patch.

Domestically produced operating systems are all based on the Linux 4.14+ kernel, which theoretically carries the risk of vulnerabilities. Users are advised to proactively contact the vendors to confirm patch status and not rely solely on this document for judgment.

Q8: Are cloud servers, virtualization platforms, and bare metal environments all at risk of this vulnerability?

A: **Yes, both face risks**, but the level of risk varies depending on the deployment model.

Deployment mode	Risk level	Reason
Bare metal servers	High	The risk is the same as with a physical machine; an attacker who gains root

		access can have complete control over the hardware.
Virtual machines (KVM/Xen/VMware)	High	Ordinary users within a virtual machine can escalate privileges to VM root, thereby attacking the virtualization platform.
Cloud servers (ECS/EC2)	High	Privilege escalation is possible within a single-tenant instance; the risk is extremely high in multi-tenant shared host scenarios.
Container services (ACK/EKS/TKE)	Extremely high	Container escape risk: A single malicious Pod can gain root access to the host machine.
Serverless/Function Computing	Low	Typically, non-persistent Linux environments have short instance lifecycles.

Special risks of cloud environments:

■ **Multi-tenant shared host**: If the cloud service provider does not patch all hosts in a timely manner, ordinary users of tenant A may escalate their privileges and attack the host, thereby affecting the instance of tenant B.

■ **Cloud vendor-hosted Kubernetes**: If the control plane (Master) node is not repaired in time, the overall security of the cluster is threatened.

Q9: Are Docker, Kubernetes and various container runtime environments affected by this vulnerability, and is there a risk of container escape?

A: **Yes, container environments are not only affected, but also face a double risk**.

Risk 1: Privilege escalation within a container. Ordinary users within a container can directly exploit this vulnerability to gain root privileges. Even if the container is configured with restrictions such as non-root users or dropping all capabilities, the vulnerability can still be exploited as long as the `AF_ALG` socket is available.

Risk 2: Container escape (Discloser claims) Because the page cache is shared between the host machine and containers, attackers can tamper with the cached copy of the host machine's setuid program (such as su) in memory. When the host machine or other

containers execute this program, the tainted cached content will be loaded, thereby triggering malicious code and gaining root privileges on the host machine (the disk files themselves will not be modified).

***Important Note**: The disclosing party (Theori/Xint) claims that Copy Fail has container escape capabilities and has announced a detailed technical article to be released. However, as of April 30, 2026, **detailed information on container escape exploitation has not been publicly released**. The community has verified that in a rootless Podman environment with user namespaces enabled, this vulnerability **cannot directly escape** to the host machine. Kubernetes 1.33+ enables user namespaces by default, which may reduce the risk of container escape.*

Affected container runtimes: Docker, containerd, CRI-O, Kubernetes, Podman (rootful mode is at higher risk).

Q10: Could embedded devices, IoT gateways, or industrial control systems that have not been updated for a long time be affected assets?

A: **Yes, and it's more difficult to repair**.

Affected device types include: routers, switches, firewalls, NAS devices, smart home gateways, cameras, industrial control systems (ICS/SCADA), vehicle systems, and medical devices.

Fixing Challenge:

• **Long kernel upgrade cycle**: Embedded device manufacturers typically release firmware updates only once every 6-12 months.

• **Static compilation limitation**: Some embedded kernels statically compile related functions into the kernel, which cannot be mitigated by unloading the module.

• **Difficulty in remote upgrades**: Industrial control systems typically prohibit remote updates and require on-site maintenance.

• **EOL devices have no patches**: A large number of IoT devices have stopped being maintained and will never receive patches.

Recommendation: Immediately take inventory of all embedded assets based on Linux 4.14+, contact the device manufacturers to obtain firmware update schedules, and implement network isolation for devices that cannot be upgraded.

III. Exploitation conditions and attack scenarios

Q11: What are the necessary prerequisites for successfully exploiting this vulnerability?

A:

Condition	Is it necessary?	Plain explanation
Local ordinary user permissions	✓ Must	An attacker needs an account on the system that can run programs.
Readable target file	✓ Must	The system contains sensitive programs (such as su) that can be read by an attacker.
AF_ALG sockets are available	✓ Must	The kernel has enabled the relevant encryption interface functions.
Root privileges	✗ Not required	The purpose of the vulnerability is to gain root access.
Network access	✗ Not required	No internet connection required, can be completed locally
Specific kernel version	✗ Not required	Linux distributions after 2017 were largely affected.
Competitive conditions	✗ Not required	No need to rely on luck, you can succeed on the first try.

Q12: Can a remote attacker exploit this vulnerability directly, or must they first gain local access to the target system?

A: **Local access must be obtained first**. Copy Fail is a purely local vulnerability and cannot be triggered directly over the network.

However, remote attackers can exploit this through the following "indirect path":

■ **Web RCE → LPE**: First, gain local command execution privileges through website vulnerabilities, then use Copy Fail to escalate privileges to root.

■ **Supply chain attack**: Inject malicious code into the CI/CD pipeline to directly obtain root privileges on nodes during the build process.

■ **Container intrusion → Escape**: First, intrude into the container, then exploit vulnerabilities to escape to the host machine.

Q13: Can a non-privileged user inside a container exploit this vulnerability to breach the container boundary and achieve privilege escalation on the host machine?

A: **The disclosing party claims it is possible, but there are significant limitations**.

Page cache is shared between the host machine and all containers, theoretically allowing an attacker inside a container to tamper with files on the host machine. However, community testing has shown that:

■ **rootless Podman + user namespace**: The root user inside the container is mapped to a non-privileged user on the host machine, **and cannot escape directly**.

■ **Kubernetes 1.33+**: User namespaces are enabled by default, which may reduce the risk.

■ **Detailed escape plan not yet disclosed**: The disclosing party announced that it would release a technical article, but it has not been made public as of April 30.

Recommendation: Do not let your guard down just because "escape may not be possible"; the situation should still be handled at the highest risk level.

Q14: What are the characteristics of publicly available PoC (proof of concept) code in terms of size, complexity, and cross-platform compatibility?

A:

Feature	Details
Volume	732 bytes (pure Python script)
Rely	Only the Python standard library is required; no additio

	nal components need to be installed.
Execution method	Python 3 poc.py, no compilation required.
Cross-platform	This applies to all affected Linux distributions and requires no modifications.
Success rate	The success rate of a single shot is extremely high, eliminating the need for multiple attempts.
Default target	/usr/bin/su can specify other setuid programs.

•**Exceptions**:

■Alpine Linux: Community feedback indicates that the PoC is not working (possibly related to differences in musl libc).

■ARM architecture (such as Raspberry Pi): The shellcode in the PoC is x86_64, which needs to be adapted.

■ Under certain strict SELinux configurations: Community feedback indicates that the PoC cannot run.

IV. Repair and mitigation

Q15: What is the official solution to completely resolve this issue, and which specific kernel security versions should be upgraded to?

A: **The radical solution is to upgrade the kernel** to a version that includes the fixes.

Kernel branch	Secure version	Illustrate
Linux 6.18	≥ 6.18.22	The upstream has released a patch.
Linux 6.19	≥ 6.19.12	The upstream has released a patch.
Linux 7.0 mainline	Includes commit a664bf3d603d	The main storyline has been fixed.
Each LTS branch	Backport (to be released)	Pay attention to the security announcements of the release version.

***To reiterate**: distributions typically fix issues via backporting, which doesn't necessarily upgrade the kernel major version number. The most reliable method is to pay attention to the distribution's official security bulletins and update your kernel packages accordingly.*

Upgrade commands for each distribution:

```
# Ubuntu/Debian
sudo apt update && sudo apt upgrade linux-image-generic && sud
o reboot
# RHEL/CentOS/Rocky/Alma/Oracle
sudo dnf update kernel && sudo reboot
# SUSE
sudo zypper update kernel-default && sudo reboot
# Arch
sudo pacman -Syu linux && sudo reboot
```

Q16: In an online production environment where business cannot be restarted or interrupted immediately, what temporary mitigation measures can be implemented?

A: The following measures can reduce the risk without patching, but **they cannot replace kernel upgrades**:

Measure 1: Disable the algif_aead module (most feasible)

```
# Prevent modules from loading automatically
echo "install algif_aead /bin/false" | sudo tee /etc/modprobe.d/disabl
e-algif-aead.conf
# Unload immediately if already loaded
sudo rmmod algif_aead 2>/dev/null || true
```

Limitations: This measure is ineffective if the kernel has this feature statically compiled (not in a module manner), and the kernel must be upgraded. Disabling modules may not work in WSL2 environments.

Measure 2: seccomp restricts AF_ALG from applying seccomp policies to untrusted containers/processes, prohibiting the creation of `AF_ALG` sockets.

Measure 3: Restrict the privileges of ordinary users, temporarily disable shell access for non-essential users, or enable maintenance mode.

Measure 4: Strengthen monitoring and response by deploying auditd rules to monitor `AF_ALG` and `splice()`, and configure real-time alerts.

Q17: Is it feasible to disable AF_ALG related kernel modules (such as algif_aead) via modprobe, and what business side effects will it bring?

A: **It is feasible and has no side effects on the vast majority of businesses**.

Business scenarios	Is it affected?	Illustrate
Web services, databases, container platforms	❌ No	Uses a user-space encryption library, without relying on AF_ALG
Disk encryption (dm-crypt/LUKS)	❌ No	Directly call the kernel crypto API, bypassing AF_ALG.
Kernel TLS (kTLS), IPsec	❌ No	Kernel native path
SSH/TLS connection	❌ No	Unaffected
OpenSSL afalg engine	□ Very few affected	Very few scenarios where AF_ALG acceleration is explicitly enabled.
Custom AF_ALG application	□ Very few affected	Embedded encryption offloading, specific HSM integration

Investigation methods:

```
ss -xa | grep AF_ALG
lsof | grep AF_ALG
```

For 99% of production environments, disabling `algif_aead` is a safe and temporary mitigation measure with no side effects.

Q18: How do I configure seccomp-bpf or customize a seccomp profile to block the exploit path of this vulnerability?

A: The core idea is to prevent `the socket(AF_ALG)` system call.

Docker custom seccomp profile: Based on the default seccomp profile, add rules to prevent socket creation with domain=38 (i.e., AF_ALG):

```
{
"names": ["socket"],
"action": "SCMP_ACT_ERRNO",
"args": [
```

```
{
"index": 0,
"value": 38,
"op": "SCMP_CMP_EQ",
"comment": "Block AF_ALG to mitigate CVE-2026-31431"
}
]
}
```

To use: `docker run --security-opt seccomp=af_alg-block.json myimage`

Kubernetes Pod configuration:

```
spec:
securityContext:
seccompProfile:
type: Localhost
localhostProfile: profiles/af_alg-block.json
```

Q19: How should container platforms adjust runtime parameters (such as seccomp, capabilities, and module loading strategies) for targeted hardening?

A:

Host machine level:

```
# Disable algif_aead on all K8s nodes
echo "install algif_aead /bin/false" | sudo tee /etc/modprobe.d/disabl
e-algif-aead.conf
sudo rmmod algif_aead 2>/dev/null || true
```

Pod security policy:

```
spec:
securityContext:
seccompProfile:
type: Localhost
localhostProfile: profiles/af_alg-block.json
containers:
- name: app
securityContext:
allowPrivilegeEscalation: false
```

```
runAsNonRoot: true
readOnlyRootFilesystem: true
capabilities:
drop: ["ALL"]
```

Enable Pod Security Standards (Restricted):

```
kubectl label namespace production pod-security.kubernetes.io/enforce=restricted
```

Q20: Can kernel Live Patching (such as Ksplice, kpatch) technology be used for online patching of this vulnerability?

A: **Theoretically possible, but not recommended as the first choice**.

The fix involves reverting `algif_aead` to an out-of-place operation, requiring modifications to multiple code sections, including scatterlist allocation, AAD copying, and sg_chain removal, resulting in a significant scope of changes. Livepatch is typically only suitable for small-scale, localized code replacements, and such structural changes may be beyond its capabilities.

As of April 30, 2026, major livepatch vendors have not released livepatch versions that address this vulnerability.

Suggestion:

■ For systems that can tolerate a brief reboot: prioritize traditional kernel upgrades.

■ For systems that absolutely cannot be restarted: Contact the livepatch vendor to confirm the patch schedule and deploy temporary mitigation measures.

■ **Do not rely on livepatch as the only fix**.

V. Testing and evidence collection

Q21: How can I quickly verify whether the currently running kernel version is within the exposure range of this vulnerability?

A: I recommend the following five-step self-check method:

```
# Step 1: Check kernel version (preliminary screening)
uname -r
# Version >= 4.14 → May be affected, continue to the next step
# Versions < 4.14 → Unaffected
# Step 2: Check kernel configuration (most accurate)
grep CONFIG_CRYPTO_USER_API_AEAD /boot/config-$(uname -r) 2>/dev/null || zgrep CONFIG_CRYPTO_USER_API_AEAD /proc/config.gz 2>/dev/null
# =y → Static compilation carries risks and cannot be mitigated by unloading the module.
# =m → Module approach, can be uninstalled to alleviate [problems].
# =n → Not enabled, absolutely safe
# Step 3: Check if the module is loaded
lsmod | grep algif
# Output present → Module loaded, risk exists
# Step 4: Test if AF_ALG is available
python3 -c "import socket; s=socket.socket(38,5,0); print('AF_ALG available')" 2>/dev/null || echo 'AF_ALG blocked'
# Step 5: Run the non-destructive testing script (recommended)
curl -sL https://raw.githubusercontent.com/rootsecdev/cve_2026_31431/main/test_cve_2026_31431.py -o /tmp/test_cve_2026_31431.py
python3 /tmp/test_cve_2026_31431.py
# [+] VULNERABLE → Vulnerable
# [-] NOT VULNERABLE → Fixed or module not loaded
```

Q22: After the system is exploited, what indicators of compromise (IoC) will be left in the system logs, audit logs, or process behavior?

A: Because the vulnerability modifies the page cache rather than the disk, traditional file integrity checks cannot detect it. We recommend paying attention to the following IoCs:

System call layer:

■ `socket(AF_ALG, SOCK_SEQPACKET, 0)`: Creates an AF_ALG socket.

■ `splice(fd, NULL, sock_fd, NULL, size, SPLICE_F_MOVE)`: Passes a file page to AF_ALG

■ Ordinary user processes frequently execute su/sudo and the process tree is abnormal

Process behavior layer:

■ Python processes create AF_ALG sockets (rarely used in normal business operations)

■A non-privileged user opens `/usr/bin/su` and then executes `splice()`

■ `The su command` failed multiple times in a short period of time, but then succeeded again.

Log check:

```
ausearch -k af_alg_socket -ts recent
grep "su:" /var/log/auth.log
grep "sudo:" /var/log/secure
```

Note: Attackers may clear logs after gaining root access. It is recommended to configure centralized log forwarding to SIEM.

Q23: Can abnormal combinations of AF_ALG and splice() be captured through eBPF probes or auditing the auditd system call?

A: **Yes**, and it is the most effective runtime detection method.

auditd configuration:

```
auditctl -a always,exit -F arch=b64 -S socket -F a0=38 -k af_alg_s
ocket
auditctl -a always,exit -F arch=b64 -S splice -k splice_call
auditctl -w /sys/module/algif_aead -pr -k algif_aead_load
```

eBPF probe (bpftrace example):

```
tracepoint:syscalls:sys_enter_socket
/args->family == 38/
{
printf("AF_ALG socket created by PID %d (%s)
", pid, comm);
}
```

Q24: Can the File Integrity Monitoring (FIM) tool detect cache-level tampering of setuid binary files?

A: Regarding this vulnerability, if the attack is successful and the system is not restarted, md5sum and dpkg can be used to discover that the /usr/bin/su file has been tampered with.

- dpkg -V util-linux
- md5sum /usr/bin/su

Q25: In memory forensics or kernel crash dump analysis, is it possible to locate traces of malicious data injected by attackers?

A: **Yes, but it's quite difficult**.

Page cached pages may be reclaimed using the LRU (Least Recently Used) algorithm, and there are currently no dedicated memory forensics plugins for Copy Fail. Recommendation:

■ If an attack is suspected, immediately freeze the system (trigger a kdump panic) to preserve the memory state.

■ Use LiME to extract the physical memory image.

■ Pay attention to whether the attacker's process (usually Python) contains strings such as `AF_ALG`, `splice`, and `authencesn` in its memory.

Q26: If the system has enabled security mechanisms or tools such as SELinux or AppArmor, can the exploit path of this vulnerability be completely blocked?

A: **The default policy usually cannot completely block it, but some enhanced configurations may limit or block its use**.

SELinux effect:

■Disabled/Permissive: No defense

■Enforcing (default Targeted policy): **Usually cannot be blocked**; the default policy does not restrict AF_ALG and splice().

■Enforcing (strict policy): May restrict process access to encrypted interfaces and setuid files.

Community testing: Some community members have reported that the PoC fails to run under certain SELinux enforcing configurations, but the default RHEL/CentOS Targeted policy usually does not prevent the vulnerability.

The root cause is that SELinux/AppArmor's access control is usually based on the VFS layer and the file system layer, while the Copy Fail write path **bypasses the VFS** and directly operates the page cache through the kernel's internal scatterlist.

Possible mitigation measures:

■ If the policy explicitly prohibits the creation of AF_ALG sockets, then exploitation can be blocked.

■ If the policy restricts the execution of su/sudo (such as confined user), privilege escalation cannot be triggered even if the page cache is tampered with.

VI. Threat intelligence and situation

Q27: To date, has this vulnerability been confirmed to exist in-the-wild exploitation?

A: As of 5 p.m. on April 30, 2026 (the date this FAQ was first drafted), **no publicly known cases of unauthorized use have been found**.

However, since this vulnerability is a local privilege escalation vulnerability and the PoC has been made public, Antiy CERT believes that it has in fact been used for attack activities, especially in cases where (1) the attacker has already obtained ordinary user privileges. (2) Malicious users within the organization.

Q28: After being publicly disclosed, has this vulnerability been included in mainstream exploit frameworks or automated attack tools?

A: It is being rapidly integrated into the community.

Platform/Tools	State
GitHub PoC	✓ Publicly available (April 29, 2026)
Exploit-DB	Pending inclusion (expected within 1 day)
Metasploit	The community module is under developmen t (expected within 1 day).
Nuclei/Nessus	The detection plugin is under development (expected within 1 day).
Automated scan ning tools	Expected to appear soon (within 1 day)

Q29: What level of real-world threat does this vulnerability pose to Critical Information Infrastructure (CII), the financial industry, and cloud-native multi-tenant environments?

A:

Industry/Scenario	Threat level	Reason
Electricity/Energy/ Transportation C II	Extremely high	The industrial control system has no t been updated for a long time, and the kernel version is outdated.
Medical CII	High	Medical device firmware update cycl e is long
Financial Core T rading System	Extremely high	Do not restart; Livepatch is not rea dy.
Online banking/p ayment	High	Web RCE → LPE chain can fully control the server.
Public cloud ECS /VM	Extremely high	Shared host machine, tenant isolatio n completely fails
Kubernetes multi-tenant cluster	Extremely high	Container escape risk; a single Pod can control the entire node.
Serverless/FaaS	Low	Short instance lifecycle, no persisten t environment

Q30: How do you assess the current response and patch release schedules of security vendors, distribution maintainers, and cloud service providers?

A:

Participants	Response sp	Current status

	eed	
Linux kernel community	⋆⋆⋆⋆⋆Extremely fast	The merge patch was released on April 1st and publicly disclosed on April 29th.
Red Hat/SUSE/Canonical	⋆⋆⋆⋆⋆Extremely fast	Security advisory has been released, and patches are being pushed out.
Debian/Fedora/Arch	⋆⋆⋆⋆Fast	Security tracker updated, patch built/pushed
Amazon Linux	⋆⋆⋆⋆Fast	Patch has been pushed out
Alibaba Cloud/Tencent Cloud	⋆⋆⋆⋆Fast	A security notice has been issued.
Domestic operating system manufacturers	⋆⋆⋆	Following upstream, backport is expected to be completed within 1-2 weeks.
Embedded/IoT manufacturers	⋆⋆Slow	Response delays, firmware update cycle 1-3 months

Assessment conclusion: International distributions and leading cloud service providers responded quickly, but large-scale patch deployments on shared host machines may take 3-7 days. Embedded/IoT is the biggest patching blind spot.