Security Vulnerability

SCTPhantom: An 18-Year-Old SCTP ASCONF Transport Use-After-Free

August 06, 2026 • 15 min read

> ***TL;DR:*** *SCTPhantom is a Linux kernel use-after-free in SCTP Dynamic Address Reconfiguration. An ordered ASCONF sequence can remove a transport and then reuse its stale pointer, leaving the association with dangling path references.*
>
> *Corvus AI developed the initial finding into a reproducible vulnerability and demonstrated local privilege escalation and container-to-host escape on the tested systems. The issue is tracked as* ***CVE-2026-64564*** *and fixed upstream by* `9b2854f86f0b`.

1. Background

Hunting for bugs in protocol code such as SCTP often means tracing dense state machines that single-pass analyzers struggle to follow. Coding agents can handle much of the mechanical work: navigating a large kernel tree, iterating on a proof of concept, building and booting a kernel, collecting sanitizer reports or panic logs, and using those results to shape the next experiment.

Coding Agent can already cover most execution loops, but a comprehensive study of kernel vulnerabilities still requires research decisions that go beyond the code-generation level, and these decisions and supporting evidence must be consistently preserved across tasks and models.

Corvus AI, developed jointly by TencentOS Security Team (Tencent Zhuque Lab) , turns this process into a persistent, multi-agent vulnerability research pipeline.

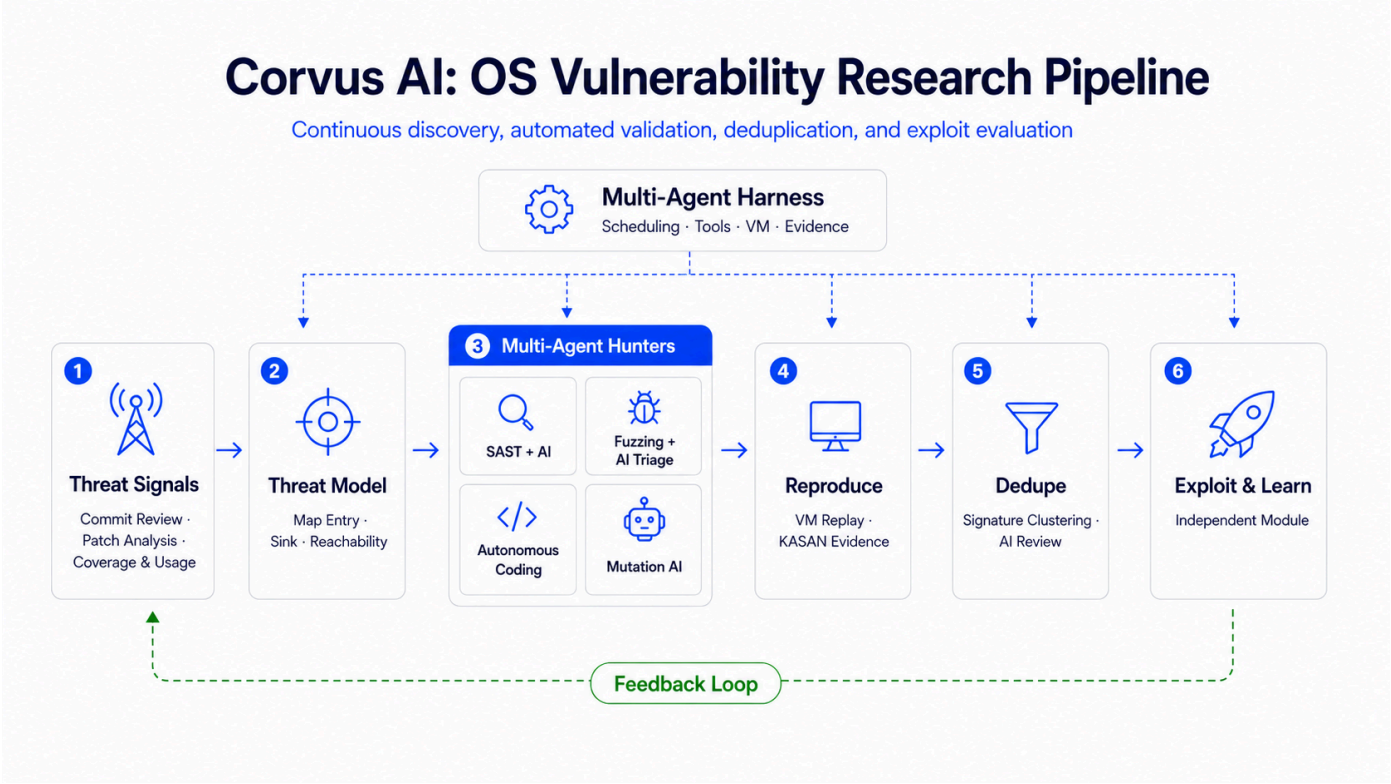

FIGURE 1 — Corvus AI: OS Vulnerability Research Pipeline.

SCTPhantom is a concrete example of that workflow. The vulnerability is in Linux SCTP Dynamic Address Reconfiguration. It arises from an inconsistency between the address used to validate a DEL-IP operation and the transport retained for subsequent ASCONF processing.

The rest of this article explains how the vulnerability was discovered and validated, how it was developed into a complete privilege-escalation chain, and how it was fixed upstream.

2. Vulnerability

2.1 Discovery

The investigation used a structured SCTP research plan to define the search space, including peer-controlled fields, ASCONF parameter ordering, multihoming state, and transport ownership. Corvus AI converted these areas into bounded source-analysis, packet-generation, crash-triage, and VM-reproduction tasks.

The issue emerged after separating the IPv4 packet source from the transport selected through the ASCONF Address Parameter. By varying peer transport states, the tests produced a reproducible case in which a later socket operation accessed a released transport. Fresh-boot reproduction confirmed the use-after-free.

2.2 SCTP and ASCONF

The Stream Control Transmission Protocol, defined by RFC 4960, is a message-oriented transport protocol with multihoming support. A single association can contain several peer paths. Linux represents the association with `struct sctp_association` and each path with `struct sctp_transport`. The transports are linked through `asoc->peer.transport_addr_list`.

Two cached pointers are important to this bug:

```
struct sctp_association

peer.transport_addr_list -> [ transport A ] -> [ transport L ] -> [ transport C ]
peer.primary_path ------------------------------^
peer.active_path  ------------------------------^
```

`primary_path` and `active_path` are expected to point to live transports owned by the association.

RFC 5061 adds SCTP Dynamic Address Reconfiguration. An ASCONF chunk contains an Address Parameter followed by operations such as ADD-IP, DEL-IP, and SET-PRIMARY. Linux processes these operation parameters in message order.

2.3 Root Cause

The vulnerable ASCONF chunk uses two different identities: the IPv4 packet source `S` and the Address Parameter `L` used to select a transport. The DEL-IP check validates the requested address against `S`, while later processing relies on the transport selected through `L`.

This allows the following ordered sequence:

```
[ Address Parameter L ] [ DEL-IP L ] [ DEL-IP 0.0.0.0 ]
```

Because `S` and `L` are different, `DEL-IP L` passes the source-address check and removes `transport(L)`. The wildcard DEL-IP then reuses the cached pointer to that transport as the path to preserve. As a result, the association can retain the removed transport in `primary_path` and `active_path`, allowing a later socket operation to dereference a stale pointer.

The upstream fix closes this identity mismatch by rejecting deletion when the selected peer is the transport retained for the ASCONF chunk.

3. Exploit Chain

The completed chain is:

```
surviving SCTP transport UAF
→ UAF #1 reclaimed by pg_vec
→ direct-map page disclosure
→ repeatable 4-byte kernel read
→ IDT-based KASLR recovery
```

```
→ UAF #2 reclaimed by controlled SCTP authentication-key data
→ controlled kernel object graph
→ data-oriented commit_creds()
→ global root
→ usermode-helper variant
→ container-to-host escape
```

Corvus AI preserved the evidence and constraints from each stage so that the chain could be developed and validated incrementally.

3.1 Surviving UAF

The exploit first needs three conditions to hold at the same time:

```
the transport has completed RCU release
AND
the association remains alive
AND
userspace can still reach the stale path
```

Deleting an ACTIVE primary preserved the association, but timers and references kept the transport from being reclaimed. Deleting an UNCONFIRMED secondary allowed release, but later protocol work destroyed the association. A confirmed ACTIVE secondary provided the required balance.

The stable setup is:

```
create a multihomed association
→ demand heartbeats until the target secondary becomes ACTIVE
→ disable heartbeats on the target and remaining paths
→ inject [Address target][DEL target][wildcard DEL]
→ allow ASCONF processing to return
→ wait for the RCU release
→ access the stale path through the surviving socket
```

The implementation blackholes the spoofed source used by the ASCONF packet so that an ASCONF-ACK cannot allocate a new transport in the released slot. CPU affinity keeps receive processing, the RCU callback, and reclaim allocations on the same per-CPU slab path during reliability-sensitive stages.

A later `getsockopt(SCTP_STATUS)` reaches the freed primary path through the live socket. KASAN confirms a slab UAF with allocation and free stacks in `sctp_transport_new()` and the RCU callback.

3.2 `pg_vec` Leak

On the representative 6.6 target, `struct sctp_transport` occupies a `kmalloc-1024` allocation. A TPACKET V1 transmit ring with 128 blocks allocates a 1024-byte `pg_vec` array:

```
pg_vec allocation

+0x000  page pointer 0
+0x008  page pointer 1
+0x010  page pointer 2
+...    ...
+0x3f8  page pointer 127
```

When `pg_vec` reclaims the transport slot, `SCTP_STATUS` interprets selected page pointers as transport fields. On this build, the returned `srtt` and `cwnd` values reconstruct one 64-bit kernel page address.

Every `pg_vec` entry also refers to a page mapped into the attacking process. The disclosure therefore gives two virtual addresses for the same physical page:

```
userspace mapping  <---- same page ---->  kernel direct-map address
```

The exploit can write fake objects through the userspace mapping and reference them through the disclosed direct-map address. All later fake objects are kept inside that exact page. This avoids an earlier, incorrect assumption that independently allocated TPACKET blocks were physically contiguous.

The leak is accepted only when the returned value is canonical and page-aligned and when data written through the userspace mapping can be consumed through the intended kernel pointer. Reclaim misses restart the victim attempt.

3.3 Arbitrary Read

`SCTP_STATUS` reports the association identifier associated with the selected transport. The relevant operation is equivalent to:

```
spinfo_assoc_id = sctp_assoc2id(transport->asoc);
```

With control of `transport->asoc`, a target word at address `T` can be read by setting:

```
transport->asoc = T - offsetof(struct sctp_association, assoc_id)
```

The returned value becomes:

```
SCTP_STATUS.assoc_id = *(u32 *)T
```

This produces a repeatable 4-byte kernel read. Controlled values for the surrounding interpreted fields keep the status path valid. The exploit uses independent victim associations for reads and rejects attempts that fail reclaim markers, canonical-address checks, or known-value validation.

3.4 IDT KASLR

The read primitive recovers the text KASLR slide from gate 0 of the fixed, read-only IDT mapping in the CPU entry area. UMIP prevents a useful userspace `sidt` result on the tested build, so the exploit reads the mapping through the kernel primitive.

A 32-bit read containing the gate's middle offset bits is enough to reconstruct the runtime address of `asm_exc_divide_error` when combined with the known low bits, canonical high bits, and target slide alignment:

```
runtime_handler = reconstruct(IDT gate 0)
KASLR_slide     = runtime_handler - link_time_asm_exc_divide_error
```

The slide is validated before it is applied to link-time symbols such as `commit_creds`.

3.5 `commit_creds`

The second vulnerable association is used to install a fully controlled transport-sized object. On the reviewed 6.6 build, `msg_msg` allocations use `GFP_KERNEL_ACCOUNT` and land in `kmalloc-cg-1024`, while SCTP transports occupy generic `kmalloc-1024`. Persistent SCTP authentication-key allocations provide controlled bytes in the matching cache.

The exploit first records the association and target transport addresses. `LIST_HARDENED` requires the reclaimed packet's empty `chunk_list` to point back to its exact list-head address, so the reclaim object is constructed for the address of the transport slot rather than as position-independent data.

The supporting object graph fits inside the direct-map page disclosed by UAF #1:

```
stale association active_path
        |
        v
[ reclaimed transport / embedded packet state ]
        |
        +---- packet->transport ------> [ fake transport ]
                                                |
                                                +---- asoc ------> [ fake association ]
                                                |                      |
                                                |                      +-- base.sk --> [ f
ake socket / cred ]
                                                |
                                                +---- af_specific -> [ fake SCTP address-
family ops ]
                                                |
                                                +---- dst --------> NULL
```

The fake address-family operations use a normal return function for `get_dst` and the runtime address of `commit_creds` for `get_saddr`. The fake association points `base.sk` at a controlled page

region containing the socket fields expected by SCTP and the fields of a privileged credential. A NULL destination stops packet construction after the credential operation.

Before the final trigger, `SCTP_STATUS` must return a marker such as `0x1337beef` from the fake association. This confirms that the intended authentication-key allocation reclaimed the dangling transport and that the controlled object graph is connected.

The final trigger is an abortive close:

```
setsockopt(SO_LINGER, { on = 1, linger = 0 })
→ close(victim_socket)
→ SCTP ABORT generation
→ output-queue flush
→ sctp_transport_route()
→ controlled af_specific->get_saddr(fake_sk, ...)
→ commit_creds(controlled_cred)
```

The chain reuses existing kernel text and does not require userspace shellcode or a traditional ROP chain. Global root is verified by comparing access to `/etc/shadow` before and after the trigger and by creating a root-owned file under `/root`, not merely by observing a changed UID.

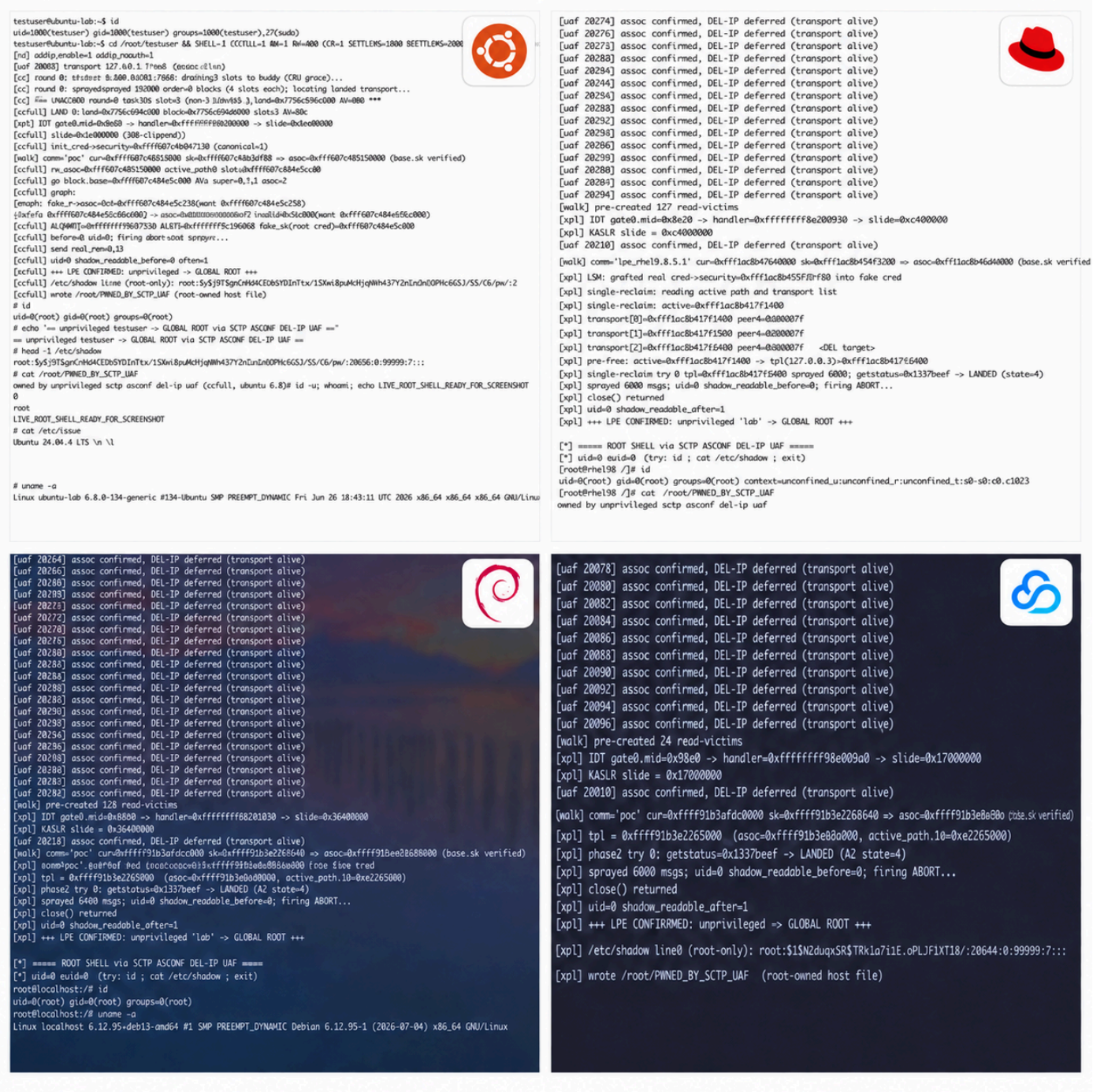

FIGURE 2. Successful local privilege escalation on four different Linux distributions

3.6 Container Escape

We then used Corvus AI to evaluate the impact of the SCTP vulnerability in container environments. The completed container chain therefore begins with the exploit command inside the container and ends with an operation in the host's initial namespaces.

Earlier PoCs enabled `net.sctp.addip_enable` and `net.sctp.addip_noauth_enable`, which made `CAP_NET_ADMIN` appear necessary. Corvus AI later found a route that leaves both sysctls

unchanged: `SCTP_ASCONF_SUPPORTED` and `SCTP_AUTH_SUPPORTED` enable the features per socket, and the trigger carries a valid AUTH chunk.

The first stages remain the same:

```
container SCTP trigger
→ UAF in the shared host kernel
→ pg_vec direct-map disclosure
→ 4-byte kernel read
→ IDT-based KASLR recovery
→ controlled kernel object graph
```

The final controlled callback invokes `call_usermodehelper_exec()` with a `subprocess_info` structure stored in the disclosed kernel-mapped page. The kernel helper runs in the initial namespaces and performs the host operation:

```
controlled callback
→ call_usermodehelper_exec()
→ process in the initial namespaces
→ host-root filesystem operation
```

FIGURE 3. Container Escape.

The retained test kept the default seccomp profile active and did not give the container `CAP_NET_ADMIN` or `CAP_SYS_ADMIN`. Six of eight attempts reached host root. The two failures stopped as clean pointer-walk misses without a kernel panic. SCTP availability, raw and packet socket access, user-namespace policy, seccomp, capabilities, and LSM policy all affect exposure in another environment.

3.7 Validation

The exploit was evaluated at different depths across several kernel and distribution targets.

Target	Kernel	Result

Research kernel	Linux 7.2-rc2	Root
OpenCloudOS-family target	6.6.119	Root
Debian 13	6.12.95+deb13-amd64	Root
Rocky Linux 9 / RHEL 9	5.14 vendor kernel	Root (SCTP module loaded)
Ubuntu 24.04	6.8.0-134-generic	Root

These results also provide a basis for assessing the impact. Under the CVSS v4.0 base metrics, the vulnerability is rated as follows:

CVSS v4.0 Base Score (CVSS-B): 8.5 (High)

Vector: `CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N`

4. Upstream Fix and Timeline

4.1 Upstream Fix

sctp: don't free the ASCONF's own transport in DEL-IP processing

```
sctp_process_asconf() caches the transport the ASCONF chunk is processed
against in asconf->transport (== chunk->transport, set once in sctp_rcv()).
For an ASCONF located through its Address Parameter by
__sctp_rcv_asconf_lookup(), that cached transport corresponds to the
Address Parameter, which need not be the packet's source address.

sctp_process_asconf_param() rejects a DEL-IP for the packet source address
(ADDIP D8, SCTP_ERROR_DEL_SRC_IP), but nothing protects asconf->transport.
A single ASCONF can therefore carry, in order:

    [Address Parameter L] [DEL-IP L] [DEL-IP 0.0.0.0]

where L differs from the source. The DEL-IP for L passes the D8 check and
calls sctp_assoc_rm_peer() on the transport that asconf->transport still
points at, freeing it (RCU-deferred). The following wildcard DEL-IP then
reuses the now-dangling asconf->transport in sctp_assoc_set_primary() and
sctp_assoc_del_nonprimary_peers(): set_primary() dereferences the freed
transport (->ipaddr, ->state) and plants the dangling pointer into
asoc->peer.primary_path / active_path, and del_nonprimary_peers(), keeping
only the pointer that is no longer on the list, removes every real
transport, leaving the association with a transport_count of 0 and
primary_path/active_path pointing at freed memory.

Reject a DEL-IP that targets the transport the ASCONF is being processed
against, mirroring the existing source-address guard, so the wildcard
branch can never reuse a freed transport.

Fixes: 42e30bf3463c ("[SCTP]: Handle the wildcard ADD-IP Address parameter")
Cc: stable@kernel.org
Signed-off-by: Jun Yang <junvyyang@tencent.com>
Acked-by: Xin Long <lucien.xin@gmail.com>
Link: https://patch.msgid.link/tencent_73762ED1DF08CC9D5F5F61954B01350CFE0A@q
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
```

The upstream patch rejects deletion of the transport retained for the ASCONF chunk:

```
        peer = sctp_assoc_lookup_paddr(asoc, &addr);
        if (!peer)
                return SCTP_ERROR_DNS_FAILED;

+       if (peer == asconf->transport)
+               return SCTP_ERROR_REQ_REFUSED;
+
        sctp_assoc_rm_peer(asoc, peer);
```

The comparison protects the object consumed by the later wildcard branch rather than only the packet-source identity. The fix changes `net/sctp/sm_make_chunk.c`. The mainline commit is `9b2854f86f0b`, and the vulnerable sequence was completed by Linux 2.6.25 commit `42e30bf3463c`.

4.2 Fixed Versions

The Linux kernel CVE announcement lists these first fixed versions:

Kernel branch	First fixed version	Stable fix
6.6.y	6.6.148	fedeb4468987
6.12.y	6.12.101	74e8f3e7114f
6.18.y	6.18.42	85aca407c560
7.1.y	7.1.6	d136b29bf91d
Mainline	7.2-rc5	9b2854f86f0b

Vendor kernels may backport the change while retaining an older base version. A version string alone does not establish exposure; the vendor advisory or source package provides the relevant fix status.

4.3 Timeline

All research dates use China Standard Time (UTC+8).

Date	Milestone
2007-12 / Linux 2.6.25	Commit 42e30bf3463c introduced the wildcard handling that completed the vulnerable sequence
2026-07-12	The ASCONF transport lifetime violation was identified and advanced from a soft-lockup to a fresh-boot post-RCU crash
2026-07-12	Private disclosure began with a readable PoC, console evidence, and a tested patch
2026-07-15	A surviving transport UAF and the first stable global-root chain were completed
2026-07-15–23	The exploit and trigger were evaluated across several 5.14, 6.6, 6.8, and 6.12-based targets
2026-07-24	The fix entered the Linux networking tree
2026-07-27	Container-to-host escape was validated
2026-08-04	The Linux kernel CVE team announced **CVE-2026-64564**

5. Conclusion

This case demonstrates the value of a persistent, evidence-driven research workflow. Corvus AI turned a protocol research template into bounded experiments, preserved negative results, and carried the same evidence chain from source analysis through fresh-boot reproduction, exploitation, and upstream remediation.

By the end, the finding came with a reusable account of the assumptions, constraints, and validation steps needed to reproduce and assess it. The conclusions rest on that record rather than on any single successful run.

References

- CVE-2026-64564 record
- Linux kernel CVE announcement
- Mainline networking-tree fix: `9b2854f86f0b`
- Introducing commit: `42e30bf3463c`
- RFC 4960: Stream Control Transmission Protocol
- RFC 5061: SCTP Dynamic Address Reconfiguration

T

Tencent Zhuque Lab

Author

Tags

AI Security Agent Skill Vulnerability MCP Jailbreak

Date Archive

2026-08	2
2026-06	2
2026-04	2
2026-03	1
2026-02	1
2026-01	1
2025-09	2