

What Bugs **Matter**?

Continuous Linux Kernel Bug Monitoring

SPEAKER

Yuan Tan

CONTRIBUTORS

Nan Li · Ren Wei · Yang Yang · Ao Zhou · Luxing Yin · Xin Liu*

bugtracker.nebusec.ai

continuous kernel bug monitoring & triage

Kernel bug reports in the AI era

Finding and reporting bugs got cheap. Verifying them did not.

- ◇ **Volume.** AI patch reviewers and LLM agents now report bugs at a scale fuzzers and humans never did.
- ◇ **Plausible, not proven.** Reports are well-argued and confident — but many describe code that no running kernel can reach.
- ◇ **Impact unclear.** A root-only corner case and an unprivileged privilege escalation read almost the same in a report.
- ◇ **The cost of checking** — reproduce, judge scope, judge severity — is still paid by hand, report by report.

The question is no longer “are there bugs?” — it’s “which of these bugs matter?”

```
llm-report.txt

# AI review of [PATCH net-next] foo: ...
"Potential use-after-free in foo_release():
 the refcount may drop to zero while the
 object is still on the pending queue."

# Looks convincing. But:
? Can a running kernel reach foo_release()?
? Can an unprivileged user trigger it?
? Has someone already reported or fixed it?
? Does it actually matter?

-----
Someone has to check – by hand.
```

Plausible $\neq$ Real

For every AI-generated report, a maintainer has to answer four questions before it is worth their time.

CHALLENGE 1

Unreachable

The bug can't actually be triggered; the path is dead.

CHALLENGE 2

Out of threat model

Only fires under assumptions no real attacker has.

CHALLENGE 3

Severity

Root vs. namespace vs. unprivileged: very different risk.

CHALLENGE 4

Unusual trigger conditions

Real, but only under odd, demanding conditions — an experienced developer has to decide if it's worth fixing.

Bug signals are **fragmented**

Many finders, many formats, no single place a maintainer can look.

— WHERE FINDINGS COME FROM

FUZZERS

syzbot

Own dashboard, own lifecycle

AI PATCH REVIEW

Sashiko, Clashiko

Pre-existing bugs buried in review replies

MAILING LISTS

People & AI agents

Many reporters send bug reports to the mailing lists

— WHAT THE MAINTAINER GETS

- ◇ **Duplicates** — the same root cause reported by several tools, in several places.
- ◇ **False positives** — plausible reports that no running kernel can trigger.
- ◇ **No priority** — a root-only corner case looks just like an unprivileged LPE.

Make the agent **crash the kernel**

- ◇ Run the agent inside **QEMU with KASAN** and kernel debug checks enabled, and let it attempt to trigger the bug for real.
- ◇ Any **kernel panic report** — a KASAN splat, a `BUG()`, or a hard panic — is our ground-truth signal that the bug genuinely fired.
- ◇ No crash, no confirmed bug.

The lesson that makes this work. A state-of-the-art agent already reproduces real kernel bugs from the patch alone, reliably enough that a reproducer is a trustworthy signal, and a persistent **failure** to reproduce is signal too.

Empirical basis on the next slide: 100 real-world KernelCTF vulnerabilities.

qemu · KASAN

```
$ ./repro_attempt_3
[ 12.441] repro: racing close() against queue
=====
BUG: KASAN: slab-use-after-free in foo_release
Read of size 8 at addr ffff888012a4c1e8
   by task repro/214
Call Trace:
  foo_release+0x1a4/0x2c0
  __fput+0x2f1/0xb70
  task_work_run+0x16a/0x260
=====
→ verified: REAL · crash log attached
```

Agents already reproduce real kernel bugs

Reproducing real kernel bugs is now something strong models do well — up to **95%** on the standard KernelCTF benchmark.

- ◇ Public studies set the floor: **over 50%** of 100 KernelCTF vulnerabilities reproduced from the patch alone, fully autonomous.

So reproduction becomes a **filter**: a bug the agent can trigger is confirmed real; one it repeatedly can't is treated as a **false positive** and dropped.

— REPRODUCTION • KERNELCTF

K-Repro • patch input

56%

GPT-5.1 Codex Max • 2026¹

THIS PLATFORM

Reproducer agent • bug-report input

95%

GPT-5.4 • 2026

¹ Baseline: Pu, Li, Liang, Cox, Wu, Shehada, Srivastav, Qian. *Patch-to-PoC: A Systematic Study of Agentic LLM Systems for Linux Kernel N-Day Reproduction*, arXiv:2602.07287 (2026).

Where sanitizers go **blind**

Some real, exploitable bugs never trip KASAN — so a crash-based oracle can't see them.

PAGE OWNERSHIP / COPY-ON-WRITE

Writing to a page you don't own

Copy Fail (CVE-2026-31431) and variants: Dirty Frag, Fragnesia

BUG	An in-place write lands on a shared page-cache page brought in by <code>splice()</code> / zero-copy.
WHY MISSED	Every access is to valid, live memory — the bug is in ownership semantics , close to a logic bug.
NEEDED	New static reasoning about page ownership — or LLMs .

STACK OBJECT LIFETIME

A pointer into a dead stack frame

GhostLock (CVE-2026-43499): futex PI, unprivileged → root, 15 years old

BUG	A task keeps <code>pi_blocked_on</code> pointing at an <code>rt_mutex_waiter</code> on a stack frame that has already returned.
WHY MISSED	KASAN has no stack-lifetime checks : use-after-scope detection was removed in 2019 (7771bdbbfd3d) for its cost and few findings.
DOWNSTREAM	GrapheneOS is looking at MTE stack tagging for use-after-scope.

As LLM-assisted bug hunting scales up, we expect **more bugs outside classic fuzzing oracles** to surface. What signal should catch them?

A crash is not always a vulnerability

Whether a trigger “counts” depends entirely on the attacker model.

— CASE · 00B WRITE IN GET_OPTIONS()

```
lib/cmdline.c · KASAN

# boot cmdline: st=0-2147483647
# range count wraps INT_MAX → INT_MIN
# ints + i now points before the array
-----
BUG: unable to handle page fault
      for address: ffffffff...
RIP: get_option+0x1a/0x90
Call Trace: get_options ← st_setup
Kernel panic - not syncing
→ verified: real 00B write
```

— REAL – BUT OUT OF SCOPE

- ◇ A genuine **out-of-bounds write**, crashing under KASAN — present since **v2.6.20**.
- ◇ It only fires if you **modify the boot command line** — not a runtime, unprivileged trigger.
- ◇ Whoever can set your boot args **already owns the machine** — so this crash **doesn't count**.

This is where verification stops. A reproducer proves a bug is **real and reachable**; it can't tell you whether it's **in scope**. Scope is a policy we set **in the prompt** — verification doesn't settle it.

Grade severity by **privilege** and **primitive**

We rate each bug on two axes: how much privilege it takes to trigger, and what capability it grants.

— WHO CAN TRIGGER IT • PRIVILEGE FLOOR

- ◇ Get a **root reproducer** first, then **step privilege down**, level by level.
- ◇ The **lowest floor** that still triggers becomes the reported severity.

L0	root / CAP_SYS_ADMIN	start	BASELINE
----	-------------------------	-------	----------

L1	user namespace	↓ step down	ELEVATED
----	----------------	-------------	----------

L2	unprivileged user	↓ step down	CRITICAL
----	-------------------	-------------	----------

— WHAT IT GRANTS • EXPLOITATION PRIMITIVE

- ◇ The same crash can grant very different power, from a **constrained write** to **arbitrary write** or **control-flow hijack**.
- ◇ Static reasoning spots the **memory corruption**; a debugger-driven check **confirms the primitive** before it counts.

IF	Invalid free	WRITE
----	--------------	-------

OUW	OOB / UAF write	WRITE
-----	-----------------	-------

CFH	Control-flow hijack	CONTROL
-----	---------------------	---------

AAW	Arbitrary write	CONTROL
-----	-----------------	---------

When verification isn't enough

Even with dynamic checks and a threat model, some bugs only fire under odd, demanding conditions.

— CASE · A TIME-RANGE PACKET MATCH

```
● ● ● cross-day contiguous match

// cross-day range backdates a day
stamp -= SECONDS_PER_DAY;

// if the clock sits near 1970:
stamp < 0 → monthday wraps > 31

val = 1U << monthday;
// out-of-range shift – undefined behavior
```

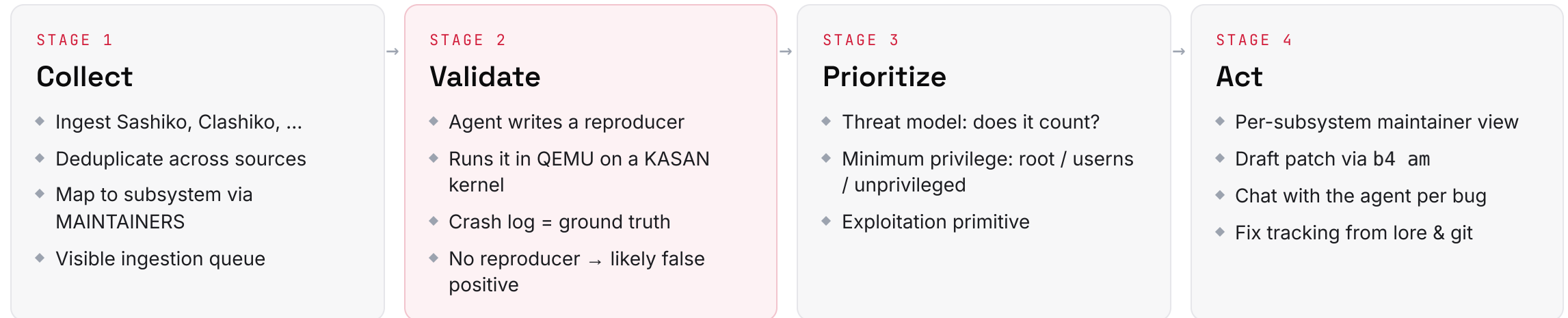
— REAL CRASH – PRACTICALLY UNREACHABLE

- ◇ It genuinely **crashes the kernel** — an out-of-range **1U << monthday** shift, real and reproducible.
- ◇ But it only fires if the **system clock sits near 1970** — a corner that **virtually never happens** in the real world.

Reality: whether bugs like this are **worth fixing** takes an **experienced developer** to judge. Verification narrows the field; it doesn't remove the expert.

How it works: from raw reports to **bugs that matter**

A continuous, fully automated pipeline for findings from any source.



Every stage runs automatically. Humans step in only where judgment is needed — bugs with unusual trigger conditions, and the final patch.

What the platform provides

Like syzbot for fuzzer bugs — an automated tracking layer for AI-reported bugs.

✓ Zero extra effort

Nothing to run, configure or maintain.
Nothing is posted to the mailing lists.

✓ Tracked automatically

Mailing lists and git are monitored; bugs are **marked fixed** automatically.

✓ Every bug reproduced

Each bug shown comes with a **reproducer** and a **KASAN crash log**.

Collect & deduplicate

Sashiko, Clashiko and more; visible ingestion queue.

Verify in QEMU

Agent writes a reproducer and runs it on a KASAN kernel.

Triage

Minimum privilege to trigger: root, namespace, or unprivileged.

Draft patch

A starting point for the fix; fetch with `b4 am`.

Per-subsystem view

Log in with a MAINTAINERS address; see only your bugs.

Chat with the agent

Discuss each bug and its fix on the bug page.

What a maintainer sees

bugtracker.nebusec.ai

Kernel Bug Tracker

Sashiko queueGuideChatsAutoYuan Tan

SUBSYSTEMS

Clear

Your name, e-mail, or st

Bugs are mapped to MAINTAINERS entries by the files they touch. Pick one or more subsystems to scope the list.

Select all

MAC80211224

NETWORKING...11624

NETFILTER1271

CRYPTO API884

NETWORKING D...501

TC subsystem458

MEMORY MANA...412

LIBRARY CODE378

NETWORKING [I...310

BLUETOOTH SU...271

NETWORKING [I...252

NFS, SUNRPC, A...243

223

Bugs in scope

194

Open

29

Fixed

All16Mine0Open2Fixed14Draft patch generated2

PoC Run Result?All194Crash2xTriggered, no crash34Couldn't reproduce57Falsified101Out of scope1

Trigger Required Privilege?All2xunprivileged0user namespace1root1

Exploitability?All2xExploit ready1Exploit very likely0Exploit likely0Not exploitable0Pending analysis1

Search bugs: id, file, description...Source: allType: allShow severity from sashiko

Crash xMAC80211 x

PICKED BY	SOURCE · ID	DESCRIPTION	TYPE	FILE	TRIGGER	STATUS
Zhiling... x	nebusec#12147	Exploit ready	use-after-free		Crashuser ns	OpenFixedOut of scope
Pick up	nebusec#12145		memory-corrup...		Crashroot	OpenFixedOut of scope

2 of 223 bugs in scope · priority order1 / 1

One bug, everything in one place

- Severity at a glance

Exploitability and the minimum privilege needed to trigger it.
- Tracked automatically

Flips to Fixed on its own once a patch appears on the mailing list.
- Root cause traced

The commit and release that introduced the bug.
- A proposed fix

With Fixes: and Cc: stable, ready for b4 am. Or chat with the agent about it.

Details blurred – this bug is still open.

bugtracker.nebusec.ai

↑

↓

12147

1 of 2 in view

Chat

×

nebusec

Exploit ready

Critical

● user namespace

Owner

Zhiling Zou

Clear

Status

Open

Fixed

Out of scope

Type

use-after-free

Found

2026-07-20 15:27 UTC (2 months ago)

Introduced by

· v6.7

PROPOSED FIX

1 patch

1 file +4 -0 +++

FIXES

Cc

stable@vger.kernel.org

Next steps

Live at bugtracker.nebusec.ai

01 Improve the quality of generated patches

02 Ingest syzbot bug reports

03 Find a way to validate driver bugs