

Sashiko

Agentic review of Linux Kernel code changes

Roman Gushchin, Google
roman.gushchin@linux.dev





LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025



What Sashiko is

Agentic harness for Linux kernel code review

Main goal: prevent new issues from entering the kernel

- Open-source code (Rust, Apache-2.0), donated to Linux Foundation
- Multi-model support: works with Gemini and all other popular LLMs
- Watches kernel mailing lists and reviews all patches
- Sends LKML-style reviews over email/patchwork when maintainers opt in
- Sashiko.dev instance sponsored by **Google**

Why a dedicated review harness?

- Quality of reviews: context management & review protocol
- Security: prompt injection
- Open-source: LLM-provider independence and full control over prompts
- Efficiency at scale: tokens and time

Core design principles

- **Full transparency:** every review, prompt, thoughts chain, tool call, and token count is public and auditable
- **Benchmark-guided:** prompt and harness changes are developed and (periodically) tested based on benchmarks
- **Strictly opt-in email/patchwork delivery:** outbound mail requires maintainers consensus
- **Kernel-like development culture:** Rust, DCO, commit log expectations

Multi-stage review pipeline

- Stages 1–7: Reviewing patch from different angles: intent, claims, control flow, resource lifetimes, locking, hardware, and security
- Stage 8: Deduplication and verification (NEW)
- Stage 9: Per-finding parallel post-verification (NEW)
- Stage 10: Report generation

Separating discovery from verification

- Asking a model to report only bugs it can 100% proof reduces the discovery rate
- Enumerating concerns first (Stages 1-7) and actively trying to falsify them against the tree (Stages 8-9) yields better results.

What flows between the stages

STAGES 1-7 · discovery · each emits this

```
{
  "concerns": [{
    "type": "Memory Leak",
    "description": "leak in ttm_bo_init() on error",
    "reasoning": "1. bo_alloc() succeeds 2. early
                  return drops the ref ...",
    "preexisting": false,
    "locations": [{
      "file": "drivers/gpu/drm/ttm/ttm_bo.c",
      "function_or_symbol": "ttm_bo_init",
      "line": 214,
      "code_snippet": "return ret;",
      "why_this_location_matters": "..."
    }]
  }],
  "dismissed_concerns": [{
    "description": "candidate we investigated",
    "reasoning": "concrete proof it does NOT apply",
    "locations": [ ... ]
  }]
}
```

STAGE 10 · verification · schema changes here

```
{
  "findings": [{
    "problem": "drm/ttm: leak in ttm_bo_init() on
               init failure",
    "severity": "High",
    "severity_explanation": "proof chain, scored
                           against severity.md",
    "preexisting": false,
    "locations": [ carried forward, refined ]
  }]
}
```

type/description/reasoning → problem/severity/severity_explanation. A concern only becomes a finding by surviving falsification against the tree, and it acquires a severity at that moment.

dismissed_concerns never reach this stage – they stay in the record as evidence of what was ruled out, and why.

1-7 discovery → 8 deduplication → 9 conflict-resolution → 10 verification → 11 report

191k

**Patch reviews completed
across 99 mailing lists**

Current throughput: ~37.8k reviews / month

19.3M

**Autonomous git tool lookups
(~105 per review)**

6.62M git_grep · 5.30M search_file_content
4.44M git_show · 1.83M read_files

Every review inspects callers, callees, lock rules and git history.

697.7B tokens processed, 83.1% served from context cache.

4.0h → 15m

p90 single-patch review turnaround time (Mar → Sep 2026)

Series: 14.7 hours → 27 minutes

How sashiko's main benchmark is built

1k real kernel regressions with known fixes (benchmarks/benchmark.json)

- Each case: the buggy commit, its later Fixes: commit, and a ground-truth problem description – spanning 314 subsystems (net 57, mm 56, rust 23, ...).
- Sashiko re-reviews the original patch with no knowledge of the fix; an LLM scores each review Detected / Partially detected / Missed.

53.6% of these bugs found with Gemini 3.1 Pro – and 100% of them had already passed human review and been merged into mainline.

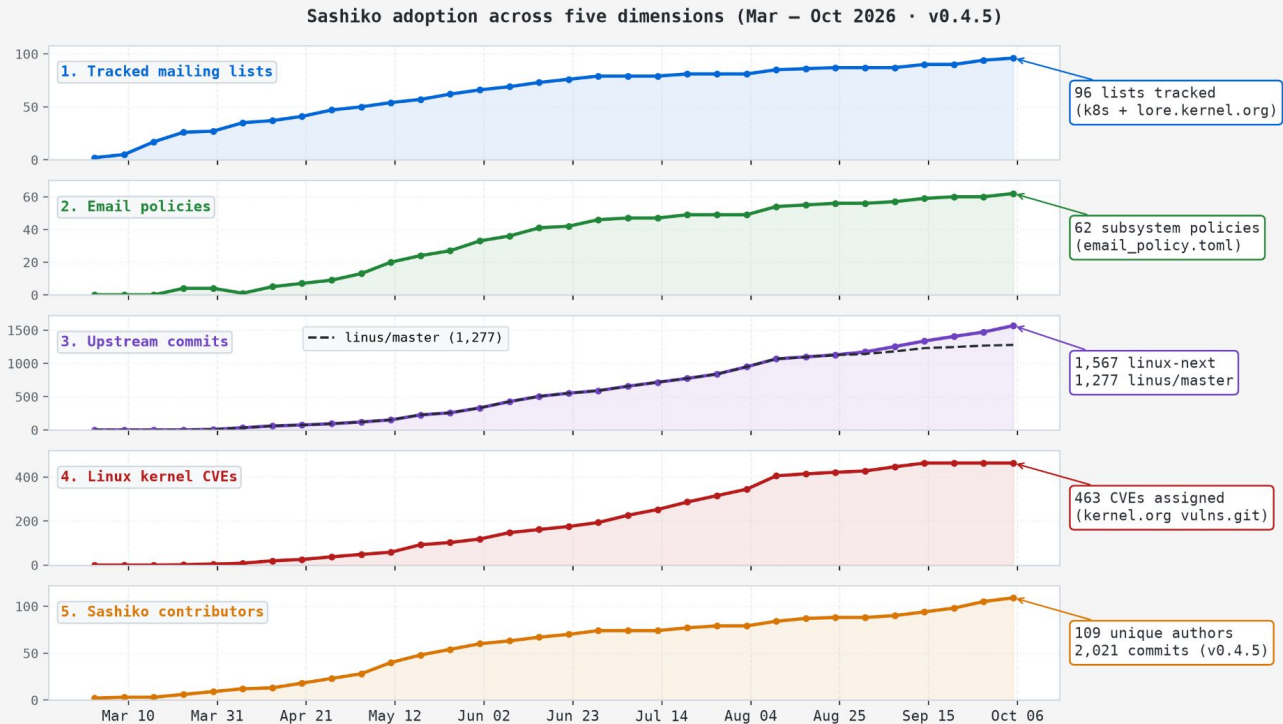
It works but there is a lot of place for an improvement!

Measuring impact in the wild

- Sashiko is only 6 months old: still early to measure real upstream impact – trends will be much easier to evaluate in 6–12 months (release cycle, LTS, ...).
- **"...fixes tag is necessary indeed, but please drop 'Reported-by: Sashiko'. If we start doing such things every patch will have it."** – Alexei Starovoitov

Straightforward fixes happen quietly in v2/v3, while email threads naturally focus on subtle cases; maintainers also ask to omit bot tags:

- Local & corporate instances run before public lists.
- Often impossible to say if the author fixed a problem because of sashiko, some other AI tools or by realizing the problem on their own



96 mailing lists · 62 maintainer email policies · 1,567 upstream commits · 463 CVEs
109 git authors and 2,021 commits contributed to sashiko.git since March 2026

7,635 replies

Direct email replies from 1,158 kernel developers

- 7,635 direct In-Reply-To: responses across 4,212 threads (plus 4,027 human emails quoting or linking Sashiko reviews).

1,277 / 1,567

**Upstream commits citing Sashiko (linus/master /
linux-next)**

1,277 merged commits in linus/master (and 1,567 in linux-next/master)
explicitly credit Sashiko via Reported-by:, Closes:, or Link: tags across
drivers (514), core kernel (210), networking (207), DRM (84), MM (75), and KVM (66).

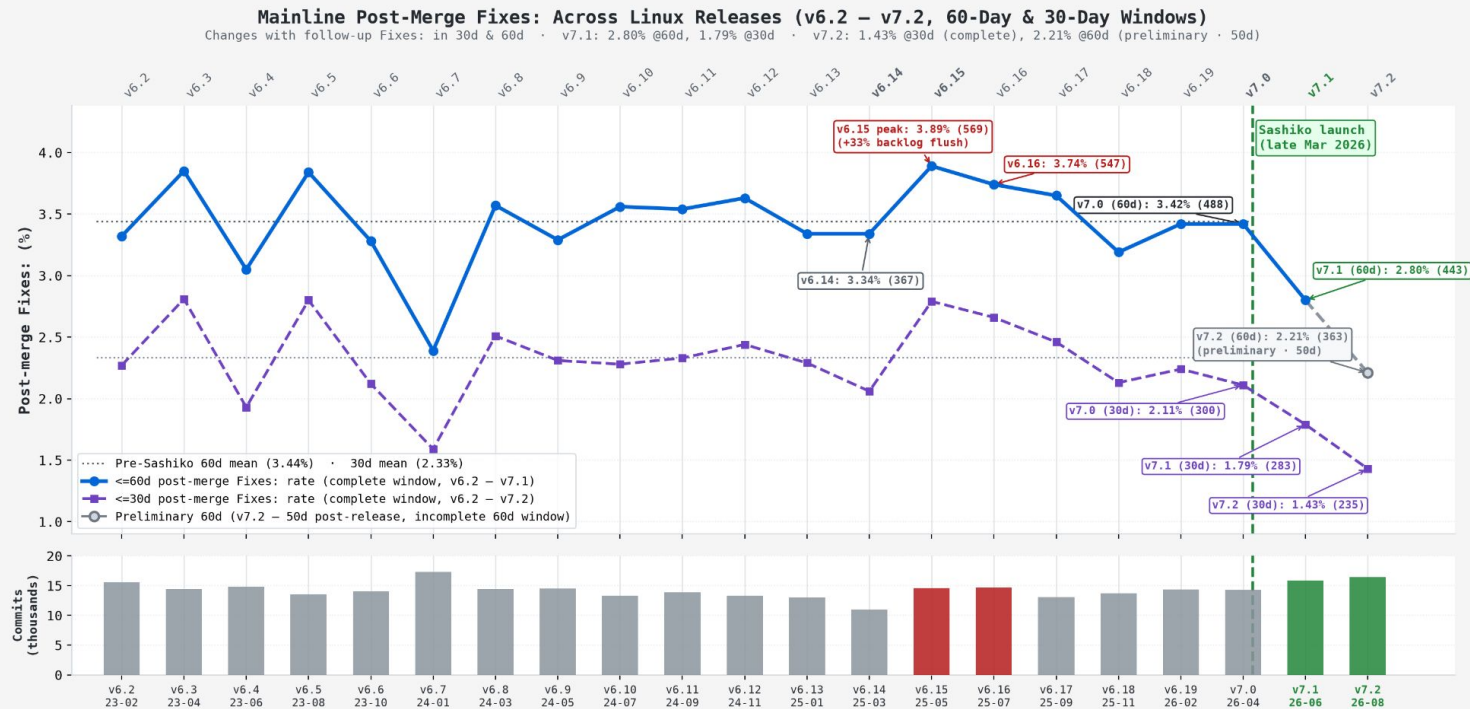
463 CVEs

2026 Linux kernel CVEs whose fix commit cites Sashiko

Cross-referenced directly against

`git.kernel.org/pub/scm/linux/security/vulns.git:`

94 in networking, 48 in block/storage, 46 in MM, 39 in filesystems, 20 in DRM,
19 in core kernel, 15 in BPF, 12 in KVM, and 170 across drivers & arch.

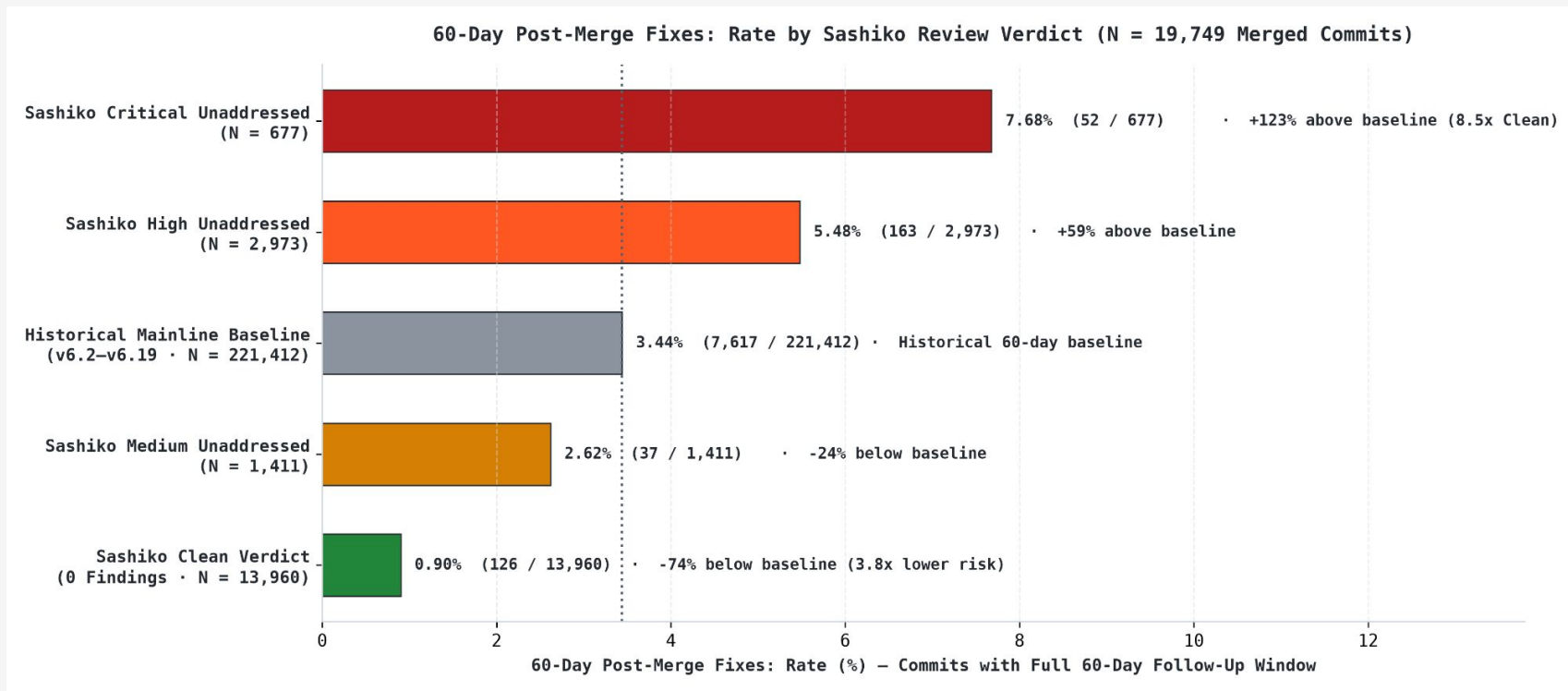


data/continuous_fixup_timeline_with_releases.csv · Pre-Sashiko 60d mean 3.44% (30d: 2.33%) · v7.1 at 2.80% / 1.79% · v7.2 at 1.43% (30d) / 2.21% (60d prelim)

Staggered Subsystem Rollout (Swimlane Matrix): 60-Day Fixes: Rate Across 15 Diverse Subsystems With Marked ★ Adoption

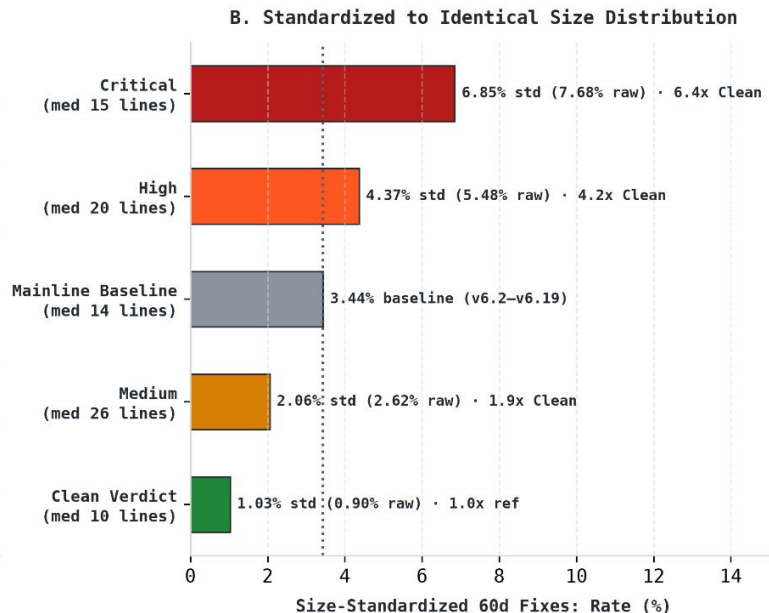
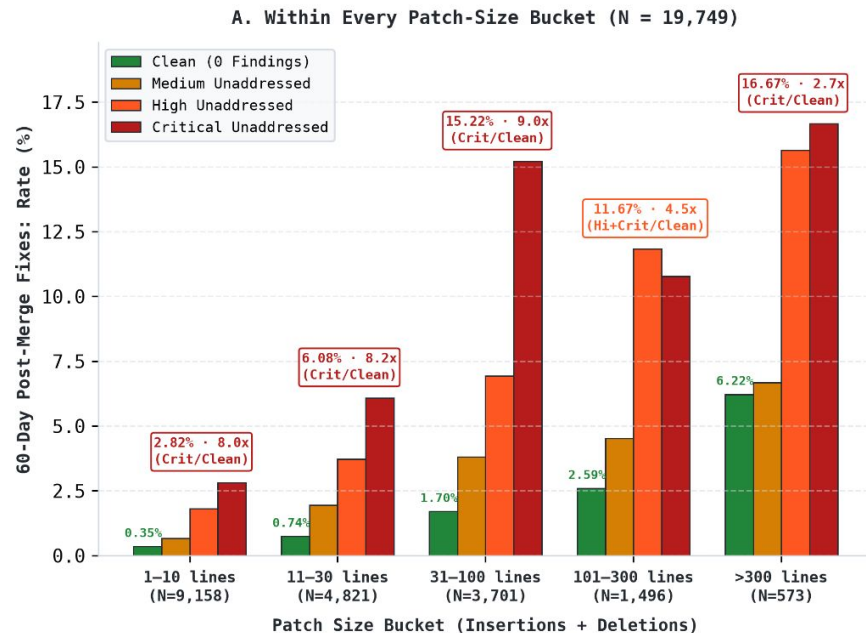
	v6.8-v6.19 Baseline (12 rel)	v6.19 (Feb '26)	v7.0 (Apr '26)	v7.1 (Jun '26)	v7.2 (Aug '26)	
Wave 1 · perf (Mar 19)	4.18%	7.20%	6.83%	★ 2.33%	1.08%	4.18% → 1.08% (-74%)
Wave 1 · mm + damon (Mar 19)	4.79%	4.51%	4.68%	★ 2.62%	2.14%	4.79% → 2.14% (-55%)
Wave 1 · net / netdev (Mar '26)	3.78%	2.51%	3.27%	★ 2.14%	2.18%	3.78% → 2.18% (-42%)
Wave 1 · hwmon (Mar 19)	1.79%	0.89%	1.47%	★ 2.94%	1.55%	1.79% → 1.55% (-13%)
Wave 1 · bpf (Mar 19)	3.92%	1.58%	3.68%	★ 2.01%	4.90%	3.92% → ★2.01% (-49% v7.1) *
Wave 2 · fs: xfs (Jul '26)	3.00%	1.72%	5.22%	1.56%	★ 0.00%	3.00% → 0.00% (-100%)
Wave 2 · kvm / virt (May 22)	2.63%	1.03%	3.55%	3.66%	★ 0.92%	2.63% → 0.92% (-65%)
Wave 2 · pci/cxl + scsi (May-Jun)	3.42%	2.74%	4.09%	2.24%	★ 1.35%	3.42% → 1.35% (-61%)
Wave 2 · drm: dri-devel (Jun 3)	5.30%	5.11%	5.97%	3.39%	★ 2.11%	5.30% → 2.11% (-60%)
Wave 2 · sched_ext (Apr 29)	8.37%	7.27%	7.41%	12.36%	★ 3.57%	8.37% → 3.57% (-57%)
No Policy · io_uring	6.72%	7.14%	7.69%	3.16%	8.62%	6.72% → 8.62% (+28%)
No Policy · core sched	5.41%	6.67%	6.15%	8.82%	8.08%	5.41% → 8.08% (+49%)
No Policy · fs: smb / cifs	3.98%	4.86%	4.66%	2.20%	8.02%	3.98% → 8.02% (+102%)
No Policy · tty / serial	4.93%	4.62%	3.33%	0.00%	6.56%	4.93% → 6.56% (+33%)
No Policy · gpio / pinctrl	5.03%	7.65%	5.97%	8.43%	5.19%	5.03% → 5.19% (+3%)

data/subsystem_adoption_timeline_v6_12_to_v7_2.csv · Wave 1 (★ v7.1) vs. Wave 2 (★ v7.2) vs. 5 Split Non-Adopters (No Email Policy)



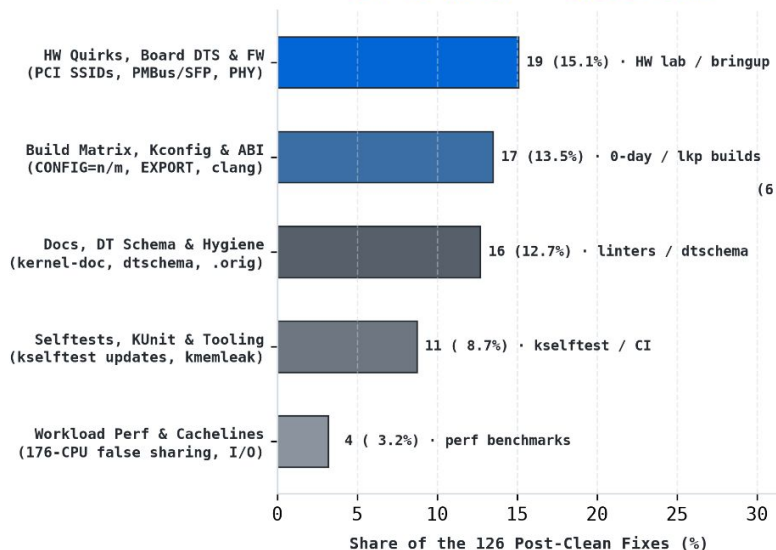
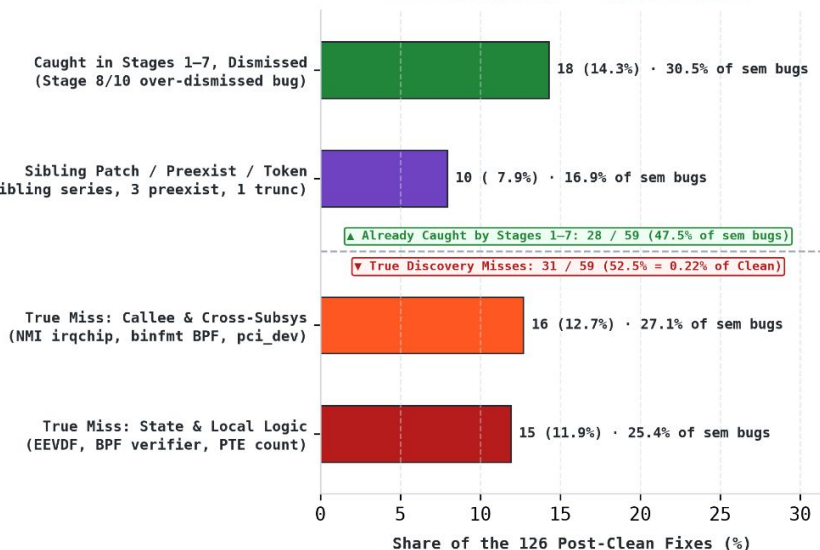
Each subsystem drops in the exact release cycle after its Sashiko policy goes live (★ at v7.1 for Mar-Apr Wave 1; ★ at v7.2 for May-Jul Wave 2), while non-adopters remain elevated. (*In bpf v7.2, 8/10 fixes were in-tree bpf-next fixups within 2-6d before merge window; mainline escapes dropped 1.41% → ★1.10% → 0.84%, -40%.)

Controlling for Patch Size: 60-Day Post-Merge Fixes: Rate by Verdict (N = 19,749)



data/patch_size_controlled_survival_60d.csv · Patch size explains ~20% of raw gap; CMH size-adjusted RR vs. Clean: Med 1.9x, High 4.2x, Crit 6.4x ($p < 1e-38$)

Inside the 0.90% 'Missed' Fixes: Auditing All 126 Post-Clean Fixes: Commits (out of 13,960 Clean)

A. Bugs Semantic Review Should NOT Target
67 / 126 (53.2%) · 0.48% of CleanB. Place for Growth for Sashiko (Semantic Bugs)
59 / 126 (46.8%) · 0.42% of Clean

What's new

Three major features landed recently:

1. Local/terminal review mode
2. Persistent bug database
3. Sashiko for Sashiko

Local review mode: \$ sashiko review

```
$ cargo install sashiko
```

```
...
```

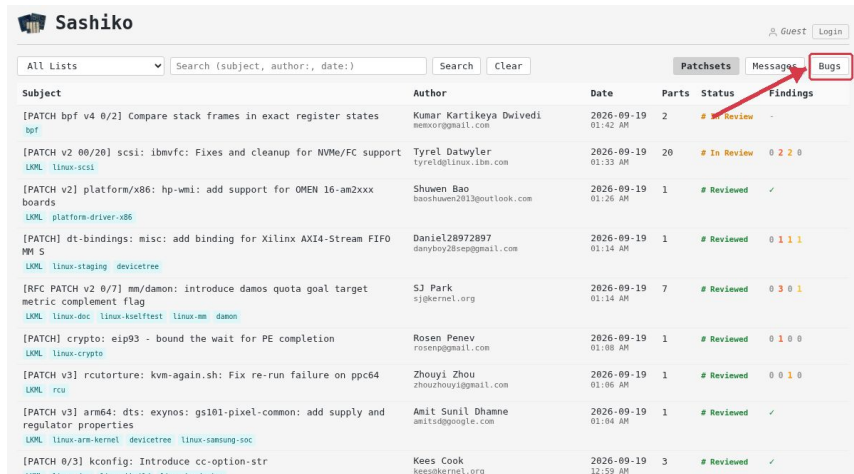
```
$ cd ~/linux && sashiko review HEAD~2..HEAD
```

- Runs directly against a local git worktree without any database setup.
- Works inside developer workflows and agentic coding loops to catch and fix bugs locally before git send-email.

9,117 open bugs

Collected in 20.5 days
(Sep 14 – Oct 5, 2026)

- Available to all in MAINTAINERS (M:/R:)
- Agentic deduplication and verification
- Stateless auth
- Place for other AI tooling to collaborate (kres, ...)
- Next: agentic upkeep (e.g. closing fixed bugs, etc)



Sashiko Guest Login

All Lists Search (subject, author:, date:) Search Clear Patchsets Messages **Bugs**

Subject	Author	Date	Parts	Status	Findings
[PATCH bpf v4 0/2] Compare stack frames in exact register states bpf	Kumar Kartikeya Dwivedi memxor@gmail.com	2026-09-19 01:42 AM	2	Review	-
[PATCH v2 00/20] scsi: ibmvfc: Fixes and cleanup for NVMe/FC support LKM linux-scsi	Tyrel Datwyler tyrel@linux.ibm.com	2026-09-19 01:33 AM	20	In Review	0 2 2 0
[PATCH v2] platform/x86: hp-wmi: add support for OMEN 16-am2xxx boards LKM platform-driver-x86	Shuwen Bao baoshuwm2013@outlook.com	2026-09-19 01:26 AM	1	Reviewed	✓
[PATCH] dt-bindings: misc: add binding for Xilinx AXI4-Stream FIFO MM 5 LKM linux-staging devicetree	Daniel28972897 danyboy289egg@gmail.com	2026-09-19 01:14 AM	1	Reviewed	0 1 1 1
[RFC PATCH v2 0/7] mm/damon: introduce damos quota goal target metric complement flag LKM linux-dc linux-kseltest linux-mm damon	SJ Park sj@kernel.org	2026-09-19 01:14 AM	7	Reviewed	0 3 0 1
[PATCH] crypto: eip93 - bound the wait for PE completion LKM linux-crypto	Rosen Penev rosenp@gmail.com	2026-09-19 01:08 AM	1	Reviewed	0 1 0 0
[PATCH v3] rcutorture: kvm-again.sh: Fix re-run failure on ppc64 LKM rcu	Zhouyi Zhou zhouzhouyi@gmail.com	2026-09-19 01:06 AM	1	Reviewed	0 0 1 0
[PATCH v3] arm64: dts: exynos: gs101-pixel-common: add supply and regulator properties LKM linux-arm-kernel devicetree linux-samsung-soc	Amit Sunil Dhanne amitsd@google.com	2026-09-19 01:04 AM	1	Reviewed	✓
[PATCH 0/3] kconfig: Introduce cc-option-str LKM linux-dev linux-kernel linux-kconfig	Kees Cook kees@kernel.org	2026-09-19 12:59 AM	3	Reviewed	✓

Sashiko for Sashiko (Sashiko self review)

- Live instance reviewing every pull request to `github.com/sashiko-dev/sashiko`.
- <https://sashiko.sashiko.dev>
- Still tuning, take it's feedback with a grain of salt.

Learning from maintainer replies

Maintainers & developers replies on mailing lists are the best signal.

reply to sashiko's feedback →

verify the explanation against the tree →

draft a review-prompt rule →

validate re-reviewing the original patch →

open a PR against review-prompts

Beyond the Linux kernel

Apparently there is [sashiko for OpenBSD](#)

Now there is sashiko for sashiko (<https://sashiko.sashiko.dev>)

Next: sashiko for kernel review prompts

? sashiko for git

? sashiko for llvm

? sashiko for qemu

The trust layer for maintainers & agents

"This is where Sashiko comes in – it basically does end up being a tool that can then increase the trust in the patch. [...] The basic issue shouldn't be AI per se, it should be that notion of 'trust'."

– Linus Torvalds

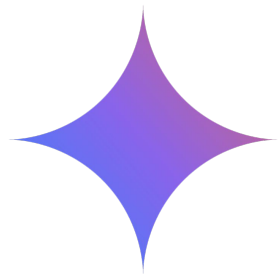
Sashiko was meant to be a tool for maintainers

An independent verification layer that increases trust in patches sent to public mailing lists and helps to prevent bugs.

But increasingly becomes a tool for agents

Coding agents call `$ sashiko review` inside their own loops.

`$ sashiko review --agent (NEW)`



Gemini 4 Argon

coming soon...

Known issues & limitations

- Large changes reviews are worse
- Keeps finding bugs even if the code is the same (v2, v3, ...)
- Does not find all bugs:
Clean pass != bug-free.
- Still hallucinates (sometimes)
- Sashiko has bugs too

Contribute to Sashiko

- Provide review feedback
- Develop per-subsystem prompts
- Contribute to Sashiko directly
- Adopt Sashiko for other OSS projects

Links & resources

- Public service: <https://sashiko.dev>
- Source code: <https://github.com/sashiko-dev/sashiko>
- Kernel review prompts: <https://github.com/masoncl/review-prompts>
- Mailing list: **sashiko@lists.linux.dev**
- Contact: Roman Gushchin <roman.gushchin@linux.dev>