git-author-guard v2 fail-closed fix (MC 104976)

Proveo FINAL Adversarial Verification — MC #104976 git-author-guard.sh v2 (fail-closed fix)

Verifier: Proveo (Angie Jones) **Date:** 2026-07-08 **Method:** Isolated scratchpad git repos only (never real repos), hook invoked exactly as Claude Code PreToolUse would invoke it (JSON piped to stdin), hook file itself NEVER modified during testing (checksum verified identical before/after all runs: `bbb4b122e5cbb8eea2376b954eff70dc3c9dc0a9396e35e426f93579db7b132c`). All repos under `/private/tmp/claude-501/-Users-makinja/a41f75a2-391f-4d27-b3db-69fdeb9cf7ac/scratchpad/proveo-104976-v2/`.

OVERALL VERDICT: **PASS**

The v1 security regression (B3/B4 — cap-at-20 silently dropping blocked-author commits from the scan while reporting ALLOW) is **genuinely closed**. Both unsafe code paths from v1 now fail closed (`exit 2`, deny) instead of capping-and-passing. I independently re-ran the full v1 suite plus 6 new adversarial scenarios (NEW-F) specifically hunting for an over-block DoS or any residual bypass introduced by the fail-closed change. **No bypass found. No unreasonable over-block found** — the one edge case that does now fail-closed (first-ever push of a big branch to a genuinely-empty remote, >100 commits) is a narrow, legitimate, override-covered edge case, not a routine workflow.

A — False-positive fix (still works)

PASS. 3 historical commits (john@/alem@alai.no/john@flowforge.alai.no) synced to remote, then 1 new `ci@flowforge.alai.no` commit. `git push origin main` → range correctly computed as `origin/main..HEAD` (1 commit only). Exit 0, `ALLOW — 1 specialist commit(s) verified.` Historical blocked-author commits correctly excluded from the scan (not re-flagged). Evidence: `scenA-stdout.log`, `scenA-stderr.log`, `scenA-exitcode.txt` (=0)

B1 — Normal block (new blocked commit, synced repo)

PASS. New `john@alai.no` commit on top of synced main. Exit 2, BLOCK, correctly identifies `0c5df792 (john@alai.no)`. Evidence: `scenB1-stdout.log`, `scenB1-stderr.log`, `scenB1-exitcode.txt` (=2)

B2 — Stale ref, fetch succeeds

PASS. Local `refs/remotes/origin/main` removed; new `alem@alai.no` commit; hook logs "Target ref ... not found locally. Fetching...", fetch succeeds (local file:// remote), range recomputed correctly. Exit 2, BLOCK, `7b6f782c (alem@alai.no)` correctly identified. Evidence: `scenB2-stdout.log`, `scenB2-stderr.log`, `scenB2-exitcode.txt` (=2)

B3 — THE v1 KILLER, re-run exactly (150 commits, oldest = john@alai.no, remote unreachable)

PASS — FIXED. Identical construction to v1 verdict (repo with 150 commits, commit #1 oldest = `john@alai.no` "THE TARGET", commits #2-150 = `ci@flowforge.alai.no` filler, remote pointed at a nonexistent path to force fetch failure deterministically).

Result: **exit 2 (BLOCK)**, not exit 0 (ALLOW) as in v1.

```
[git-author-guard] Target ref refs/remotes/origin/main not found locally. Fetching
origin/main...
[git-author-guard] Fetch failed or ref still missing. Falling back to HEAD --not --
remotes=origin
[git-author-guard] BLOCKED — degenerate scan (150 commits, target ref unreachable). Cannot
verify safely.
```

Confirmed fail-closed-by-construction, not scanned-and-passed: I traced the exact code path. The BLOCK fires at the `if [ "$FALLBACK_COUNT" -gt 100 ]` check (`git-author-guard.sh:251`, exit at line 283) which executes **before** the per-commit author-classification loop (`COMMIT_LINES` loop starts at line 385, well after the block returns). This means the block is unconditional on count (>100) — it does NOT depend on first scanning all 150 commits and finding `john@alai.no`; it blocks the entire degenerate condition regardless of what authors are present. This is genuine fail-closed behavior, not a lucky scan result.

I also independently re-verified the malicious repo was constructed correctly: `git log --format="%H %ae" | tail -1` → oldest commit = `john@alai.no`, confirmed.

Evidence: `scenB3-stdout.log`, `scenB3-stderr.log`, `scenB3-exitcode.txt` (=2)

B4 — v1 killer #2: cap-attack via detached HEAD / undeterminable branch (blocked alem@alai.no)

PASS — FIXED (2 variants tested).

Variant 1 (v1-identical repro): Detached HEAD, 150 commits, oldest = `alem@alai.no`, push command `git push origin HEAD:main`. Branch parser resolved `PUSH_BRANCH="origin"` (same parsing quirk noted in v1 verdict), landed on the same fetch-fail→degenerate-count path as B3. Exit 2, BLOCK: "degenerate scan (150 commits, target ref unreachable)". Evidence: `scenB4-stdout.log`, `scenB4-stderr.log`, `scenB4-exitcode.txt` (=2)

Variant 2 (true "cannot determine branch" path, added by Proveo for completeness): Detached HEAD, no refspec at all (`git push origin`), only 2 commits (oldest = `alem@alai.no`). Because the commit count was small (<100), the code took the fetch-fail→full-author-scan branch (not the count>100 branch) and correctly classified and named the blocked commit: exit 2, `BLOCKED ... - d1fa2da6 (alem@alai.no)`. This confirms the small-range fallback path still does full per-author classification (not just count-based fail-closed) — belt-and-suspenders. Evidence: `scenB4b-stdout.log`, `scenB4b-stderr.log`, `scenB4b-exitcode.txt` (=2)

C — Stateful (3 consecutive invocations)

PASS. Repo synced with 1 allowed commit already pushed; 3 consecutive hook invocations with no new commits between them (identical to v1 test). All 3 → exit 0, "No pending commits in push

range — ALLOW." No flake, no drift, no state leakage between invocations. Evidence: `scenC-invoke1-stderr.log`, `scenC-invoke2-stderr.log`, `scenC-invoke3-stderr.log`, `scenC-summary.txt` (all exit=0)

D — Input contract intact (PreToolUse stdin JSON)

PASS. All 5 sub-cases:

- D1: `tool_name != "Bash"` → exit 0 (no interception)
- D2: Bash, non-`git push` command → exit 0 (no interception)
- D3: malformed JSON on stdin → exit 0, degrades safely (no crash/hang)
- D4: empty stdin → exit 0, no hang/crash
- D5: valid JSON + blocked push → exit 2, well-formed JSON stdout, `hookSpecificOutput.permissionDecision == "deny"` confirmed No syntax break found that would disable all pushes or all protection. Exit 0/2 semantics correct throughout. Evidence: `scenD1..D5-stdout.log`, `scenD1..D5-stderr.log`

E — Override token (single-use, mode/mtime enforced)

PASS. Full sequence against a stable blocked push (RANGE_SHA-derived token path):

- E1: no token → exit 2, BLOCK, override path printed
- E2: valid token (mode 0600, fresh mtime) installed → exit 0, `ALLOW_OVERRIDE`, token consumed (confirmed via stderr "OVERRIDE TOKEN consumed")
- E3: immediate retry, same push → exit 2, BLOCK again (single-use enforced — token file confirmed gone: `ls` reports "No such file or directory")
- E4: token re-installed wrong mode (0644) → exit 2, BLOCK, "INVALID (... mode=0644 ...). Ignoring." — mode check enforced
- E5: token re-installed correct mode but stale mtime (120s, >60s TTL) → exit 2, BLOCK, "INVALID (age=120s ...). Ignoring." — TTL check enforced Override logic block is untouched by the v2 diff (confirmed via diff against `.bak-20260708` in the v1 verdict — byte-identical); re-confirmed behaviorally here. Evidence: `scenE1..E5-stdout.log`, `scenE1..E5-stderr.log` (E1/E3/E4/E5=2, E2=0)

NEW-F — Adversarial: hunting for a NEW bypass or over-block DoS introduced by fail-closed

PASS across all 5 sub-tests. This was the core focus of the final verification — does fail-closed overcorrect into blocking ALL large/normal pushes (DoS on legit work), or does any path still silently ALLOW while a blocked commit is unscanned?

F1 — reachable-but-empty remote (edge case discovered, documented, not a regression)

150 commits, all allowed authors, remote is a real reachable bare repo but has **zero refs** (nothing ever pushed there — true first-push-of-a-branch scenario). Fetch legitimately fails (`fatal: couldn't find remote ref main`, exit 128 — verified manually, not a network flake). Hook result: exit 2, degenerate-scan BLOCK. This is a genuine edge case introduced in reachability by the fail-closed change (v1 would have capped-and-allowed here too, just unsafely) — it is override-covered and rare (first push of >100 commits to a brand-new remote branch). Not a routine workflow. Documented as expected/acceptable fail-closed behavior, not a defect. Evidence: `scenF1-stdout.log`, `scenF1-stderr.log`, `scenF1-exitcode.txt` (=2)

F1b — reachable remote WITH existing branch, 150 new legit commits (the real over-block test)

PASS — no over-block. Remote seeded with an initial commit (so `origin/main` truly exists on remote), local tracking ref dropped to force the fetch-path, then 150 new `ci@flowforge.alai.no` commits added locally. Fetch **succeeds** (remote genuinely has `main`), range computed correctly, all 150 commits scanned individually. Result: **exit 0, "ALLOW — 150 specialist commit(s) verified."** No cap, no false block. This directly disproves an over-block DoS on the standard/reachable/large-push case — the >100 degenerate-block only fires when the fetch itself fails or the ref genuinely cannot be resolved, not merely because the count is large. Evidence: `scenF1b-stdout.log`, `scenF1b-stderr.log`, `scenF1b-exitcode.txt` (=0)

F2 — exact boundary (100 vs 101 commits, unreachable remote, oldest = blocked)

- **100 commits** (not `> 100`): NOT degenerate-blocked; falls through to full per-author scan → correctly identifies and BLOCKs on `john@alai.no` by name (`0b3ca5bf`). Exit 2.

- **101 commits**: `> 100` → degenerate fail-closed BLOCK fires (count-based, before classification). Exit 2. Both correctly BLOCK; there is no gap at the boundary where a blocked author could slip through unscanned in either direction. Evidence: `scenF2-exactly100-{stdout,stderr,exitcode}.log/.txt` (=2), `scenF2-exactly101-{stdout,stderr,exitcode}.log/.txt` (=2)

F3 / F3b — stale-ref mixed scenarios

F3: repo1's own single legit commit against its own (locally-existing, potentially stale) tracking ref → exit 0, ALLOW (correct; no cross-actor commits involved). F3b: constructed a scenario where a **second actor/clone** pushes a blocked-author commit directly to the shared remote (repo1 never fetches it, never has it in its own HEAD ancestry). repo1 then pushes its own unrelated legit commit → exit 0, ALLOW. This is **architecturally correct, not a bypass**: this hook gates a single invocation's own outgoing commit range (`origin/<branch>..HEAD` from the invoking repo's perspective). A commit that was pushed by a different actor/repo was never part of *this* push's range — it would have been (and, per this hook's design, should be) gated by the hook running on *that other actor's own* push invocation, not retroactively by every other clone's subsequent unrelated push. This is the same scope boundary the hook has always had (author-guard is a per-push gate, not a full-history auditor) and is unrelated to the v1/v2 diff. Evidence: `scenF3-stdout.log`, `scenF3-stderr.log`, `scenF3-exitcode.txt` (=0), `scenF3b-stdout.log`, `scenF3b-stderr.log`, `scenF3b-exitcode.txt` (=0)

NEW-F conclusion: No new bypass found. The fail-closed change does NOT introduce a blanket over-block on normal large pushes (F1b proves 150 legit commits sail through when the remote/ref is genuinely resolvable). The only new fail-closed trigger is a genuinely unresolvable/unreachable target (network failure, misconfigured remote, or a from-scratch first push of a large branch to an empty remote) — narrow, override-covered, and strictly safer than the v1 behavior it replaces (which silently ALLOWed in the same conditions).

Summary Table

Scenario	Result	Notes
A — false-positive fix	PASS	Unchanged, reproduced independently
B1 — normal block path	PASS	
B2 — stale-ref fetch-succeeds path	PASS	
B3 — cap-attack (150 commits, oldest=blocked, unreachable)	**PASS (FIXED)**	exit 2, confirmed fail-closed-by-construction (count check precedes author scan)

Scenario	Result	Notes
B4 — cap-attack via detached HEAD path	**PASS (FIXED)**	exit 2, both variants (v1-repro + true undeterminable-branch)
C — stateful (3x consecutive)	PASS	No flake
D — hook input contract	PASS	No JSON/exit-code contract break
E — override token (valid/reuse/bad-mode/stale)	PASS	Unchanged logic, all sub-cases correct
NEW-F1 — empty-remote first-push edge case	PASS (documented edge, not a defect)	Narrow, override-covered
NEW-F1b — reachable remote, 150 legit commits	**PASS — no over-block**	Full scan, all ALLOWED, not capped
NEW-F2 — exact 100/101 boundary	PASS	No gap either side of the threshold
NEW-F3/F3b — stale-ref / cross-actor mix	PASS	Correct scope boundary (per-push gate, not full-history audit) — unrelated to this diff

OVERALL: **PASS** — security regression is genuinely closed, without introducing an

over-block DoS on legitimate work. MC #104976 may be closed as done.

Rationale:

1. Both v1-unsafe paths (detached/undeterminable-branch cap, and >100-commit degenerate fallback cap) now fail closed (exit 2, deny) instead of silently capping-and-ALLOWing. Verified via direct code trace (block precedes author-classification loop) — this is not a coincidental scan result, it is unconditional on the degenerate condition.
2. Re-ran the exact v1-killer constructions (B3, B4) independently — both now correctly BLOCK.
3. Actively hunted for the failure mode NOT yet tested by the builder (a new over-block DoS on ordinary large pushes) — found none: F1b proves 150 legitimate commits against a genuinely reachable/resolvable remote are fully scanned and ALLOWED, not capped or falsely blocked.

4. Boundary-tested the exact >100 threshold (F2) — no gap in either direction.
5. All previously-passing scenarios (A, B1, B2, C, D, E) remain PASS — no regression introduced elsewhere in the diff.
6. Hook file checksum confirmed unchanged throughout testing (`bbb4b122e5cbb8eea2376b954eff70dc3c9dc0a9396e35e426f93579db7b132c`) — findings reflect the deployed v2 hook, not a modified test copy.

Evidence paths (all under /Users/makinja/system/evidence/104976/proveo-verify-v2/)

scenA-*, scenB1-*, scenB2-*, scenB3-*, scenB4-*, scenB4b-*, scenC-*, scenD1..D5-*, scenE1..E5-*, scenF1-*, scenF1b-*, scenF2-exactly100-*, scenF2-exactly101-*, scenF3-*, scenF3b-*

Test harness scripts (reproducibility) in scratchpad: `/private/tmp/claude-501/-Users-makinja/a41f75a2-391f-4d27-b3db-69fdeb9cf7ac/scratchpad/proveo-104976-v2/` scenA.sh, scenB1B2.sh, scenB3-retest.sh, scenB4-retest.sh, scenB4b-truedetached.sh, scenC.sh, scenD.sh, scenE.sh, scenF1-large-legit-reachable.sh, scenF1b-large-legit-existing-remote.sh, scenF2-boundary-100-101.sh, scenF3-stale-existing-ref.sh, scenF3b-stale-ref-fastpath.sh

---

Revision #1
Created 2026-07-08 12:13:16 UTC by John
Updated 2026-07-08 12:13:16 UTC by John