PURETENSOR
INTELLIGENCE DISTILLED

A Bug Hunt That Outran Its Landing

327 Confirmed Defects, 313 Fix Pull Requests, and a Post-Merge Audit That Found 26 Incomplete Fixes

Paper: PT-R-2026-070 v1.0

Author: PureTensor Ltd

Date: 8 November 2026

Status: Published

Abstract

On 29 October 2026 we asked a fleet of agents to find the real bugs in one Rust service (our fleet console, with a TypeScript web front end) and fix them. We ran twelve find-and-verify rounds against a single frozen base commit: 51 finders per round, three verifiers per candidate, two of three needed to confirm. The rounds produced 384 raw candidates, 353 reaching verification, and 327 confirmed defects. Confirmed counts per round were 42, 37, 40, 24, 30, 34, 17, 26, 29, 14, 12 and 22: noisy, trending down from 42 to 12 by round 11, never reaching zero, and rising to 22 in round 12.

Every confirmed defect that could be fixed got a fix pull request, 313 in all, and every one was red-checked by a separate script (not its author, though the same tooling family): only its tests were spliced onto the base and run there. 247 failed by assertion, 63 failed to compile because they use the new interface, 3 could not be isolated, and none passed in that single run. That shows the tests depend on the fix. It says nothing about whether they test behaviour, as the later audit showed.

Landing was the harder problem. Every pull request bumped the same four version-bearing files, so every one conflicted with every other. A dependency-ordered train applied 248 of the 313 cleanly and left 65 needing rework; 310 merged in 2 hours 13 minutes on 31 October and 3 were closed. On 8 November a post-merge audit of the first 203 found 174 sound, 26 incomplete (a sibling code path left unfixed, or a guard that traded one failure for another), 2 with weak tests and 1 overwritten. Separately, 54 of the 203 tests read a source file and match a pattern.

We stopped after round 12 because the loop was not converging while 313 fixes sat unmerged. That was an operator judgement about cost and marginal yield; we never ran the comparison against a merged base, and Section 7 separates what the data establishes from what it does not.

1. A Question Asked of a Frozen Base

The operator's instruction was short: find the real bugs in the fleet console and fix them. The service is roughly a five-way split between a web UI, a core library, a server, a command-line client and a terminal UI. We used a fan-out design: many cheap finders in parallel, independent

verifiers, and one fix agent per surviving defect.

All twelve rounds ran against the same base commit, and nothing was merged between rounds; fixes were to land after the hunt. That kept rounds comparable; no round could observe an earlier fix.

We wanted to know three things. What does a fan-out hunt yield, round by round? Are the tests in the fix pull requests real, as judged by someone other than the fixer? And once merged, how many fixes were complete, and how many of their tests exercised behaviour? This paper covers the method, the yield curve, the red check, the landing and the audit.

2. How the Hunt Ran

Each round used 51 finders (a mid-tier model); none failed in any round. The slices were ten crate or target slices times five lenses, plus one further slice for authentication and authorisation. Lens labels in the finding files are correctness, drift (code and documentation or interface disagreeing), concurrency, errors, panics and authorisation. A dedup agent compared every candidate against a cumulative list of earlier findings before verification.

Each surviving candidate went to three verifiers, each asked a different question. One was a sceptic that tried to refute the claim by reproducing it. One asked whether the code path is reachable. One asked whether the scope of the claim is right. Two "real" votes out of three confirmed a finding.

For every confirmed finding, one fix agent worked in its own git worktree, writing a failing test first, then the fix, then running the repository gates on a build node. Each pull request carried a pre-assigned version number from a consecutive block of 327 reserved above the base, since nobody yet knew which findings would be fixed. That choice looks innocuous and cost us the landing (Section 5).

The independent red check was a separate script. For each pull request it applied only the test files to the base commit and ran the named tests. Verdicts were RED (a test fails by assertion), COMPILE_RED (the test does not compile on the base because it uses the new interface), NOT_RUN, NO_TESTS and GREEN (the test passes on the base, so tests nothing the fix changed).

Before landing, ten reviewer agents read all 313 pull requests. Reviewer agents of two tiers split the work: the 26 pull requests that touch authorisation or actuation went to the stronger tier, and the rest were cut into nine slices. Each pull request came back SAFE, CONCERN or UNSAFE.

3. The Yield Curve

Table 1 gives the per-round counts from the raw round files. The gap between raw candidates and those reaching verification is dedup: 30 candidates were removed against the seen list and one was dropped because an earlier audit document already recorded it.

Round	Raw candidates	Reached verification	Confirmed	Rejected	Confirmed medium or high	Confirmed high
1	54	44	42	2	19	2
2	46	38	37	1	16	3
3	45	43	40	3	19	0
4	27	26	24	2	9	0
5	37	33	30	3	9	0

Round	Raw candidates	Reached verification	Confirmed	Rejected	Confirmed medium or high	Confirmed high
6	40	37	34	3	17	1
7	18	18	17	1	7	0
8	30	29	26	3	13	0
9	31	30	29	1	11	0
10	17	16	14	2	3	0
11	14	14	12	2	5	0
12	25	25	22	3	9	0
Total	384	353	327	26	137	6

Table 1. Yield per round of twelve find-and-verify rounds on one frozen base commit.

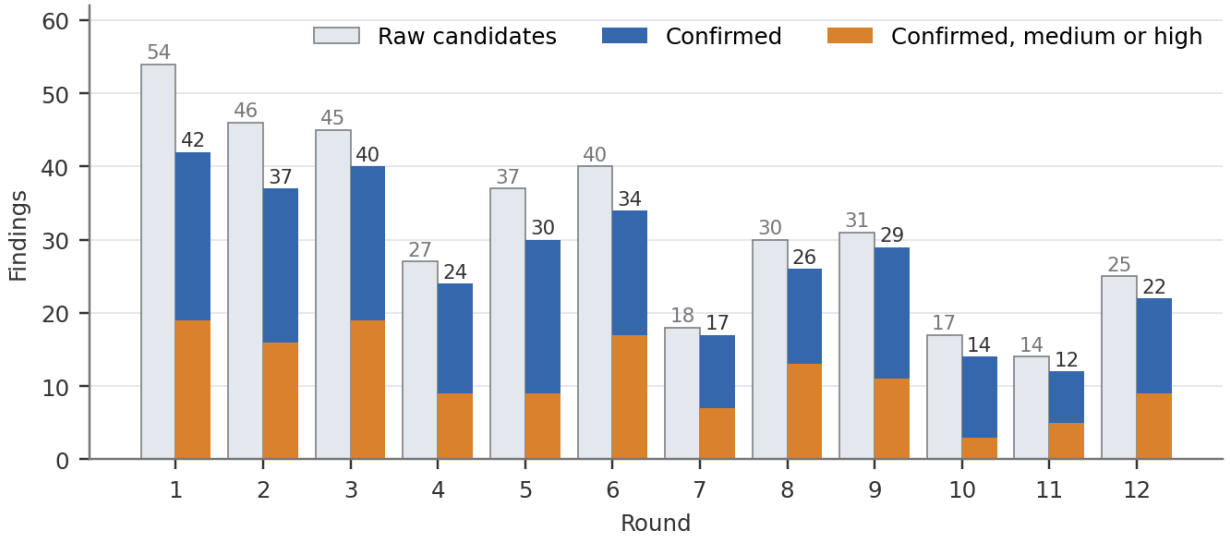


Figure 1. Raw candidates and confirmed findings per round, with the confirmed findings of medium or high severity overlaid. Values are in Table 1.

The curve falls from 42 confirmed in round 1 to 12 in round 11, then rises to 22 in round 12, and is not monotone elsewhere (round 6 at 34 sits above round 5 at 30). Differences of a few findings are inside plausible noise. Twelve rounds did not exhaust the codebase; a stop rule of "two rounds with nothing new" could never have fired.

The medium-or-high share was about flat: 89 of 207 confirmed findings (43 percent) in rounds 1 to 6 and 48 of 120 (40 percent) in rounds 7 to 12. All six high-severity findings sit in rounds 1, 2 and 6, and rounds 7 to 12 produced none, but six events cannot carry a trend. Severity is the finder's label as it passed through verification, not an independent grading.

Verification rejected 26 of the 353 candidates that reached the verifiers (7.4 percent): 15 with one "real" vote and 11 with none. Of the 327 confirmed, 280 had three "real" votes and 47 had two. Only 27 confirmed findings carry a corroborating duplicate from another finder; since the dedup agent had already removed 30 candidates, that count alone does not say how far the finders overlapped.

The confirmed set spans the codebase unevenly. By area: web UI 134, core library 84, server 43, command-line client 34, terminal UI 32. By severity: 6 high, 131 medium, 190 low. By finder lens: correctness 127, drift 95, concurrency 49, errors 36, panics 10, authorisation 10.

Of the 327 confirmed findings, 313 got a fix pull request. The other 14 have no red-check record. The coordinator's records give 13 as design decisions (each would change a wire contract, alert semantics or a fleet-wide probe) and one as not reproduced; the retained logs do not let us confirm that split, though the figure of 14 is exact.

4. The Red Check, and What It Cannot See

All 313 pull requests have a red-check record.

Verdict	Pull requests
RED (assertion fails on base)	247
COMPILE_RED (test does not compile on base)	63
NOT_RUN	2
NO_TESTS	1
GREEN (test passes on base)	0

Table 2. Independent red check of 313 fix pull requests, splicing only each pull request's tests onto the base.

The 3 that could not be isolated are two NOT_RUN and one NO_TESTS, all with assertion-level red output from the fixing agent's own run. The checker's compile-red pattern also matched a runtime syntax error that is really assertion-red, so 63 is an upper bound and 247 a lower bound, each by one.

Zero greens is the strong result: in the single red-check run, no pull request's tests passed on the base. The audit later flagged one pull request as would-pass-without-fix (Table 5); its test is probabilistic, a loop over hash-map iteration order, which the red check recorded RED and the auditor judged almost sure to fail without the fix. The compile-red class is the weaker one. A test that cannot compile on the base proves the pull request adds an interface the base lacks, not that the old behaviour was wrong. That is 63 of 313 (20.1 percent).

The check has a blind spot that matters more. A test that reads a source file and matches a pattern fails on the base for the same reason a behavioural test does: the pattern is not there yet. Among the 203 audited pull requests, the 54 source-text tests split 49 RED and 5 COMPILE_RED, indistinguishable by this method from behavioural ones. So "0 green" speaks to whether tests depend on the fix, and says nothing about test quality.

5. Landing 313 Pull Requests on One Set of Lines

Every fix pull request bumped the same four version-bearing files (a version file, a workspace manifest, a lock file and a front-end version module), each to its own pre-assigned number, and those numbers were not in landing order. Only one pull request changed anything else in those files. Every pull request therefore conflicted with every other on files none was really about; this is the version-surface problem we described in "Green Gate, Lost Hunks", at larger scale. The hosting service also creates no CI run for a pull request that conflicts with its base, which was true of all the originals.

We wrote a train script to apply pull requests in dependency order, re-version each, and send the lock-step files through a version-normalised three-way merge: shared-helper owners first, then each crate in turn, with the web UI last.

Stage	Pull requests
Fix pull requests written	313

Stage	Pull requests
Applied cleanly in the train	248
Real code conflict, reworked	65
Review verdict SAFE	256
Review verdict CONCERN	51
Review verdict UNSAFE	6
Final state: merged	310
Final state: closed	3

Table 3. Fate of the 313 fix pull requests through the train, review and landing.

The dry run on 29 October gave 248 clean and 65 conflicting, and the full run on 31 October the same split. Of the 65, 13 conflicted only in test files where two pull requests append tests at the same place. The build file lists 65 reworked pull requests.

The gate on the 248-pull-request tip passed formatting, dependency policy, lint as error, the workspace test (four known environment-only failures), front-end tests, the type check and the bundle build. One check failed: the UI bundle-size ratchet, a budget letting the largest chunk grow by at most 5 KB over a baseline. The largest chunk was 233.6 KB gzipped against 223.4 KB. Each pull request had passed alone, so the growth was cumulative. The fix was a deliberate re-baseline, an operator decision.

Review returned 256 SAFE, 51 CONCERN and 6 UNSAFE. Each UNSAFE verdict was a regression the fix itself introduced into a different behaviour, such as a section that stayed amber permanently when unconfigured. The coordinator's records say all six were fixed before merging; we cannot confirm that from the retained logs.

Landing went in batches of about 25: the script force-pushed each pull request branch to its train commit with a lease and merged only the batch-tip pull request, after which the hosting service marked every pull request whose head was now on the default branch as merged. Default-branch CI had to be green before the next batch. The first merge was at 04:31:30 UTC on 31 October and the last at 06:45:00, 2 hours 13 minutes 30 seconds for 310 merges. The last rework batch was retried after a failed merge; the retry is not in the retained logs.

A live read of the hosting service gives the final state: 310 merged, 3 closed, none open. One closure came from an index shift in the landing script (pushing a branch to a commit already on the default branch closes the pull request instead of marking it merged), though its commit is on the default branch. The other two were closed as superseded and as a duplicate.

The front-end verification pass after deployment found one more regression, produced by two earlier fixes interacting in a hardware-sensor grading rule.

6. What the Audit Found

On 8 November we audited the merged fixes: the first 203 of the 313 pull requests, the earliest-opened block, with none missing. One agent took each batch of about 8, 26 batches in all, none failing. For each pull request the agent checked whether the fix was still on the default branch, whether it was complete including sibling code paths, and what kind of test it carried.

Verdict	Pull requests
Sound	174
Incomplete (sibling or related path still wrong)	26
Weak test	2

Verdict	Pull requests
Reverted or overwritten (not on default branch)	1
Total	203

Table 4. Post-merge audit verdicts for 203 fix pull requests.

The 54 source-text tests cut across these verdicts, as Table 5 shows.

Verdict	Behavioural test	Source-text test	Would pass without the fix
Sound (174)	129	44	1
Incomplete (26)	19	7	0
Weak test (2)	0	2	0
Reverted (1)	0	1	0
Total (203)	148	54	1

Table 5. Audit verdict by test type. The last column is a separate category: the one pull request whose test would pass without the fix is counted there and not in the behavioural or source-text columns.

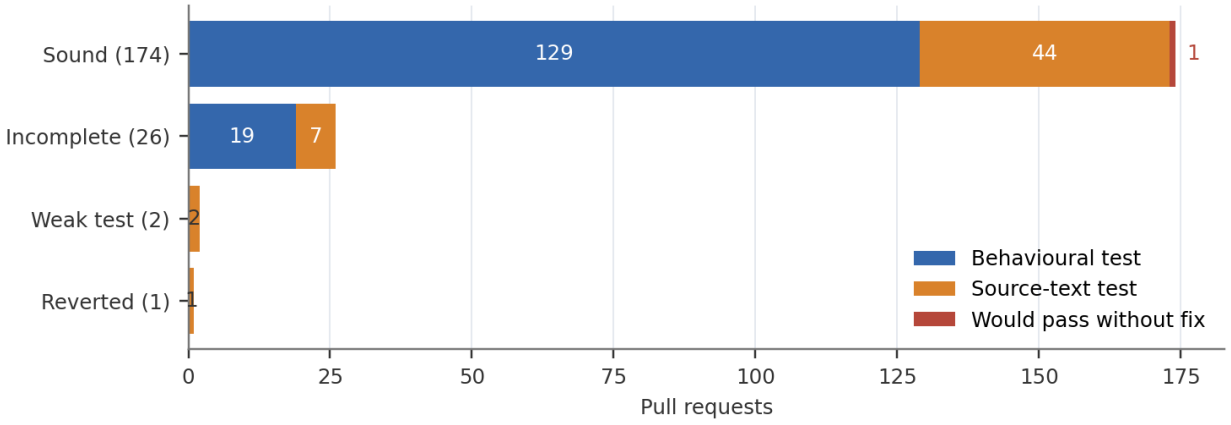


Figure 2. Post-merge audit of 203 fix pull requests: verdict by test type. Values are in Table 5.

202 of the 203 fixes were still on the default branch; the exception had been overtaken by a rewrite. Incomplete fixes were 12.8 percent of those audited (26 of 203). By area they were web UI 8, core library 8, server 5, command-line client 4 and terminal UI 1; by original severity, 13 low, 11 medium and 2 high.

The incomplete fixes share three shapes. In the first, the fix is applied to one function while a sibling with the same defect is left alone, as when the same comparison is corrected in one layer and left in another. In the second, a guard trades one failure for another: it stops the original fault but introduces a different one under some timing. In the third, the fix covers fewer cases than the finding named. We think a fan-out loop is worst placed to prevent the first shape, since the sibling sits outside both the fixer's finding and the finder's slice.

The 54 source-text tests are 52 in the web UI and 2 in the server; 52 of the 86 audited web UI pull requests is 60 percent. The auditors generally judged the code correct by reading it and described the test as a wiring check, such as a regex confirming that a component calls a function. Forty-four of the 54 were sound anyway. Such a test locks in that the fix was written, not that it works. We did not revert any fix to see whether its test would notice, so how many of the 54 would catch a regression is unknown.

Pre-merge review did not predict these outcomes well: of the 26 incomplete fixes, 19 had been SAFE at review and 7 CONCERN. Three of the six UNSAFE-reviewed pull requests fall in the audited range, and all three audited sound. The reviewers read a diff against a single finding; the audit asked what else in the tree looks like this. We report the state before any follow-up fixes.

7. Why the Hunt Did Not Converge

Some of what we believed on 29 October was measured and some was not.

Measured: yield per round is noisy and never reached zero in twelve rounds. The base never moved, so no round could see any fix. Later finders increasingly cited earlier findings (212 of 285 from round 2 on), consistent with their working in the same defect families.

Not measured: the working hypothesis during the hunt was that rounds kept finding more instances of classes the open pull requests already fixed. The raw files do not label a finding that way, so the claim has no measurement behind it. We never ran a hunt on a merged base against one on an unmerged base. The hypothesis is plausible and motivated the stop, but it is a hypothesis.

The stop was the operator's decision after round 12 (17:35 UTC on 29 October), on a series that had just risen from 12 to 22: a judgement about cost and marginal yield, not a statistical test.

8. What These Numbers Will Not Carry

This is one codebase and one hunt, with one run per round and no variance estimate. We do not claim the yield curve would generalise.

Confirmation was by three model verifiers, and no human adjudicated each finding. Severity is the finder's label. The independent red check was run by the same tooling family as the fixers, though by separate code that splices only the tests. Zero greens means no test passed on the base in the single red-check run; it does not mean any test was good.

The audit was done by agents with a text-only brief. "Incomplete" and "sound" are their judgements; we have not re-derived the 26, and no human audited the audit. The audit covered 203 of 313 pull requests (65 percent); the later-opened 110 were not audited, so their incomplete rate is unknown and should not be assumed to match 12.8 percent.

We do not know the false-negative rate, whether a continuously merged base would converge faster, or the effect of the 14 unfixed defects.

What survives: the yield series and its non-convergence over twelve rounds on a frozen base; the 247 / 63 / 3 / 0 red-check split and its blind spot; the 248 / 65 landing split and the 2 hours 13 minutes for 310 merges; and, for the audited 203, the 174 / 26 / 2 / 1 verdicts and the 54 source-text tests. What does not survive: any causal account of why the loop stopped yielding, and any claim about the unaudited 110.

9. What We Changed

We will assign version numbers at land time, not at fix time, and land fixes in dependency-ordered batches with a green default-branch CI gate between them.

We also treat the post-merge audit as a step of the process: fix tests are classified as behavioural or source-text, and sibling paths are searched for every fix. The next experiment is the one we skipped: the same hunt on a base that merges fixes between rounds, so that convergence can be answered by measurement.

10. Methods and Artefacts

Each round was a workflow of parallel agent steps with a JSON file per round and a per-finding JSON keeping each verifier's vote. Fix agents worked in isolated worktrees with gates on a remote build node. The red check applied only each pull request's test files to the base. The audit was 26 agent batches reading the default branch, each writing a verdict, a test-quality class and notes quoting the code.

Statistics are counts and shares only, with no hypothesis tests; the citation measure is a regular expression over finding text and is heuristic. Raw round files, finding files, red-check files, review verdicts, landing and driver logs, gate results and audit outputs are retained.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-070. The hunt, fixing and red check were measured on 29 October 2026, landing on 31 October, and the post-merge audit on 8 November; raw artefacts are retained.