PURETENSOR
INTELLIGENCE DISTILLED

Red-Teaming Your Own Benchmark

Three model lineages attack a code-scored bank

Paper: PT-R-2026-013 v1.0

Author: PureTensor Ltd

Date: 24 August 2026

Status: Published

Abstract

A benchmark scored entirely by code, with no language-model judge, promises objectivity: either the checker accepts the response or it does not. That promise holds only if the checkers themselves are correct. We report an adversarial audit of our internal frontier benchmark, a 94-item bank spanning ten dimensions, in which three model lineages (a GPT-5.6-family seat, Gemini 3.7 Flash, and our on-premises Qwen3.8-27B) were asked to construct responses that the checkers would accept while being wrong, or to show that an item was itself defective. Every claimed exploit was re-executed through the real checker code; no attacker's self-report was trusted.

The audit produced 217 false-green exhibits: responses that were wrong yet passed. Every one of the 94 items was false-greened by at least one lineage; 32 items were false-greened by all three. The exhibits collapsed into four classes. Two classes carry organic risk (they would bite honest models): unchecked tool arguments, where the right tool called against the wrong target passed, and confidence-ignored calibration items. Two classes are adversarial-only: assertion inversion, where text containing the gold keyword while asserting the opposite passed, and JSON fields the checker does not inspect. Fixes shipped for the organic-risk classes; the adversarial-only residue is documented but not patched, because closing it would require redesigning assertion checking.

A separate finding concerns non-blind inflation. The bank's author-answered self-baseline scored 99.2 on the agentic-weighted profile. A fresh, blind Claude Opus instance given only the question template scored 94.7 under the same checkers. The gap of about 4.5 points is the measured cost of the author knowing what each item wanted. That figure is a direct measurement, not an estimate, and it is the reason self-baselines must never be quoted as ceilings.

One item, D4-04, had failed all 18 stored strong-model runs. Reading it revealed that the intended opening user turn was missing; every model's clarifying non-call was arguably correct, yet the author's self-baseline had passed because the author knew the unstated key. This is the exact non-blind trap the bank's own design document warns against, found in the wild. The audit's blind-ceiling phase now serves as a standing gate: no new item ships until a non-author model solves it blind.

We bumped the bank from version 1.0 to 1.1. Pre-audit stored scores are not comparable to post-audit runs. The regression gate grew from 69 gold and 23 wrong cases to 83 gold, 30

wrong, and 6 calibration cases, all green. The 190 residual false-green exhibits are retained as the seed corpus for a v1.2 sweep.

1. The Instrument Under Test

Our internal frontier benchmark contains 94 original items distributed across ten dimensions: D1 tool selection, D2 abstention, D3 tool-argument construction, D4 multi-step execution, D5 reasoning, D6 false-premise detection, D7 instruction compliance, D8 structured output, D9 grounding in supplied sources, and D10 calibration. Every verdict is produced by code. The harness employs 23 check types; there is no language-model judge. Because the items are original, training-set contamination is impossible.

Three composite profiles are reported from any run: a flat (unweighted) mean, a scorer-weighted profile, and an agentic-weighted profile (the weightings are fixed in the harness and emphasise, respectively, answer quality and tool-loop behaviour). The highest score ever recorded on the bank, 99.2 agentic, 98.8 scorer, 98.5 flat, was the author-answered self-baseline. That run was not blind: the author knew what every item wanted and answered from the emitted question template without opening the checkers or gold data. The bank's design document states plainly that a non-blind self-baseline is a known trap, because the author can pass an item that no stranger could.

Before the audit, a checker regression gate (the "anti-false-red gate") ran before any score was trusted. A set of 69 known-gold answers had to pass; a set of 23 known-wrong answers had to fail. If either condition broke, the run was rejected. The gate's purpose is to catch checker regressions, but it cannot catch checkers that were wrong from the start.

2. Audit Design

We recruited three model lineages as attackers and blind-ceiling providers. Lineage A was a frontier hosted model from OpenAI's GPT-5.6 family at high reasoning effort. Lineage B was Google's Gemini 3.7 Flash at high reasoning effort. Lineage C was our on-premises 27-billion-parameter open-weight seat (Qwen3.8-27B, FP8) at medium effort; its real benchmark run also served as its ceiling. A fourth seat, a fresh headless Claude Opus instance given only the question template with no checker access and no memory of the bank, provided an additional blind ceiling.

The audit proceeded in four phases. Phase 0, cohort mining, gathered statistics over 18 stored strong-model runs of the bank. Phase 1, blind ceilings, had each seat answer all 94 items blind, scored by the identical checkers. Phase 2, the attack proper, asked each of the three attacker lineages, per item, to construct a response that the checkers would accept while being wrong, or to show the item itself was defective. This produced 282 attack units (3 seats times 94 items). Every exhibit was re-executed through the real checkers; no attacker's claim was trusted on its word. Phase 3 walked the harness validity findings.

Total wall time was about two hours. The local attacker ran on a two-GPU compute node whose GPUs peaked at 64 and 74 °C at a 325 W power envelope during an 8-concurrent attack batch of 16 minutes. Idle temperatures at close were 36 and 38 °C. No GPU errors occurred. The production seat kept serving throughout.

3. Phase 0: Cohort Mining

One item, D4-04, failed all 18 strong stored runs. Reading it revealed that the checks demanded a specific case key in a tool call, but no message in the item ever stated the key; the intended

opening user turn was missing. Every model's clarifying non-call was arguably correct. The author self-baseline had passed it because the author knew the unstated key. This is the exact non-blind trap the design document warns about, found in the wild. We fixed D4-04 before the attack waves; Lineage C independently re-flagged it from the cohort statistics.

Of the 94 items, 34 passed all 18 strong runs at single-shot temperature 0. These items are dead weight at that setting, though several still discriminate under five-rollout sampling at temperature 0.7. Twenty-nine items sit in the discriminating band, where some strong models pass and others fail.

Unscorable clusters appeared in D6, D7, and D9: up to 12 of 18 runs on one D6 item exhausted the token budget before answering. This is a harness budget-calibration gap; those dimensions are under-measured on several models. We logged it as a v1.1 follow-up.

4. Phase 1: Blind Ceilings

The blind ceilings were scored by identical checkers under the original bank (v1.0) and again after the fixes (v1.1).

Seat	v1.0 checkers	v1.1 checkers
Fresh Opus (blind)	94.7	100.0
Lineage B, Gemini 3.7 Flash high (blind)	94.7	97.9
Lineage A, GPT-5.6 family high (blind)	85.1	91.5

Fresh blind Opus and Gemini tied at 94.7 under v1.0. There is no gross Claude-family advantage in the blind channel. Opus was in fact the worst of the three on D9 (grounding), the phrasing-sensitive dimension.

The author-answered self-baseline (99.2) sits about 4.5 points above fresh same-family Opus (94.7). That gap is the measured non-blind inflation. It is a direct measurement, not an estimate, and it is the reason self-baselines must never be quoted as ceilings.

All three ceilings "failed" three D10 items (D10-03, D10-04, D10-08) while stating a probability of 0.25: the unknowable stratum behaving exactly as documented. Those items are now scored on calibration rather than on the letter chosen.

5. Phase 2: The Attack

Every one of the 94 items was false-greened (a wrong response accepted by the checkers) by at least one lineage. The audit produced 217 exhibits in total; 32 items were false-greened by all three lineages.

Classification collapsed the 217 exhibits into four classes.

Class C1, unchecked arguments, covered about 40 items. The right tool with the wrong target passed. One exhibit from Lineage A called the correct storage-health tool against the wrong node with a date range in 2020; another injected an attacker-chosen attendee into an unchecked calendar field. This class carries real organic risk: an honest model making a plausible mistake could hit it. We fixed D1 (identity-argument pins on items D1-01 to D1-06); D3 and D4 partial-argument items remain for a v1.2 sweep.

Class C2, assertion inversion, covered about 30 items. Text that contains the gold keyword or number while asserting the opposite passed (example: "10, but Rule A was applied only 9 times"). This is a structural limit: surface checks verify the presence of correct content, not the absence of contradiction. Organic risk is low against honest models; the class is adversarial only. Contradiction traps are the v1.2 direction. We documented the class but did not patch it.

Class C3, confidence ignored on D10: the gold letter with a stated probability of 0 passed. We fixed this: knowable items now require a stated probability above the base rate; unknowable items score the calibrated probability and ignore the letter.

Class C4, JSON spot-checks, covered 6 items. Fields the checker does not inspect are free. Documented.

Eleven items drew multi-lineage disputes. Three were the D10 unknowables (fixed). Eight items had keyword lists that were too tight, each with runnable correct-answer exhibits from two seats: D2-12, D5-01, D6-04, D6-08, D7-04, D9-01, D9-07, D9-08. All eight were confirmed by hand against the exhibits, and the keyword lists were widened. Those eight items had been suppressing roughly 5 correct points per ceiling seat.

The D6-04 exhibit exposed a checker-primitive bug: no Unicode normalisation, so a correct answer containing "per-protocol" with a U+2011 non-breaking hyphen was invisible to the substring check. We fixed the bug in the shared containment primitive for the whole bank.

Twelve single-lineage disputes were adjudicated. D4-04 was confirmed (already fixed). The rest were C2-class or attacker misfires (one example: Lineage C submitted a correct refusal as its "wrong answer" on D2-09).

6. Phase 3: What the Audit Says About the Harness

The bank had no item-answerability gate. D4-04 shipped because no non-author had ever solved the bank blind. The audit's own Phase 1 is that gate. New standing practice: a blind solve by a non-author model before any new item ships.

Budget starvation (the D6/D7/D9 unscorable clusters) remains open. Per-dimension token budgets for prose dimensions need raising, or per-item budgets need to be introduced.

One cosmetic issue surfaced: one check type printed its failure note even on pass, misleading during adjudication but harmless to scores.

Comparability is broken. The bank version was bumped from 1.0 to 1.1. Pre-audit stored scores are not comparable to post-audit runs (16 items' checks changed, plus the D10 semantics). Historical runs stand as v1.0 records.

7. Fixes Shipped

Bank v1.1 includes the following changes: D4-04 rebuilt; Unicode normalisation in the containment primitive; eight keyword lists widened; identity-argument pins on D1-01 to D1-06; D10 unknowable stratum and the above-base-rate rule.

Every fix carries a paired regression. The audit added 14 gold regressions that must pass, 7 wrong regressions that must fail, and 3 calibration regressions. The gate now holds 83 gold, 30 wrong, and 6 calibration cases, all green.

Post-fix, the false-green exhibits fell from 217 to 190. The residue is the documented C2 and C4 classes, which are adversarial-only and cannot be closed without redesigning assertion checking. The 190 remaining exhibits are retained as the seed corpus for the v1.2 sweep.

8. Context from the Surrounding Landscape

Public benchmark hygiene work in 2026 has shown large defect rates in established banks. One study found roughly 43% of benchmarks saturate within 24 months of release. A verified re-issue of a well-known expert-level exam flagged 74.4% of its items. A frontier mathematics benchmark corrected 42% of its problems after review.

Our own design document had anchored two claims on that literature that the audit could not verify, and we withdrew them. One premise about a graduate-level science benchmark being roughly 8% defective was likely false. One citation about an instruction-hierarchy benchmark was unverifiable. We found errors in our own framing while looking for errors in our items.

9. What These Numbers Will Not Carry

The audit has several structural limits.

Sample size. The bank contains 94 items. The attack produced 282 attack units (3 lineages times 94 items). The blind ceilings are single runs at temperature 0. No rollout variance is captured. A single run can land anywhere in a model's distribution; the ceilings are point estimates, not confidence intervals.

Single-author bank. The bank was written by the same agent lineage that operates the fleet. The audit tests the checkers, not the item content. If the items themselves are biased toward a particular reasoning style, the audit cannot detect that bias.

Non-blind inflation measurement. The 4.5-point gap between the author self-baseline (99.2) and fresh blind Opus (94.7) is a single comparison. It is not a confidence interval. It is not generalisable to other banks or other authors. It is a direct measurement of one author's advantage on one bank.

Adversarial-only classes undocumented in production. The C2 (assertion inversion) and C4 (JSON spot-checks) classes are adversarial-only. We have no evidence that honest models hit them in production. We have no evidence that they do not. The 190 residual exhibits are retained, but their organic risk is unknown.

Budget starvation. The D6/D7/D9 unscorable clusters are a known gap. Those dimensions are under-measured on several models. The audit does not fix this; it only documents it.

Comparability. Pre-audit stored scores are not comparable to post-audit runs. Any trend analysis across the version boundary is invalid.

Attacker calibration. Our working rule is that worker self-assessment sits around 70% calibrated. The audit re-executed every claimed exploit through the real checkers, so self-report error does not propagate to the final counts. But the 70% figure is itself an estimate, not a measurement.

Claims that survive: the 217 false-green exhibits are real (every one was re-executed). The four-class taxonomy is descriptive (it is a classification of observed exhibits, not a prediction). The 4.5-point non-blind inflation is a direct measurement. The D4-04 defect is confirmed (18/18 failures, plus the missing user turn). The Unicode bug is confirmed (the exhibit is reproducible). The eight tight keyword lists are confirmed (the widened lists now pass the exhibits).

Claims that do not survive: any assertion about organic risk for C2 and C4 classes, any assertion about budget-starvation impact on dimension scores, any assertion about the generalisability of the 4.5-point figure.

10. What We Changed, and What Comes Next

We changed two practices. First, no new item ships until a non-author model solves it blind. The audit's Phase 1 is now a standing gate. Second, every checker fix carries a paired regression: a gold answer that must pass and a wrong answer that must fail. The gate that cannot go red is the recurring fleet failure class; a regression pair per fix is how a checker earns trust.

The next experiment is the v1.2 sweep. The 190 residual false-green exhibits are the seed corpus. The C1 partial-argument items (D3 and D4) will receive identity-argument pins. The C2

class (assertion inversion) will be addressed with contradiction traps: items whose gold answers contain a keyword that, if negated, should fail. The budget-starvation gap will be addressed with per-dimension or per-item token budgets. We will report the results when the sweep is complete.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. Measurements for this paper were taken on 21 August 2026; raw artefacts are retained. This paper is PT-R-2026-013.