Who Caught It: Keeping a Public Score on the Human in the Loop

How one practice records, for every logged miss in its AI-assisted work, who caught it, and what an audit of that register found about who writes the record down. Set against what labs, researchers and regulators are doing about human oversight right now, newest first.

Berg Atkinson, Wolfberg LLC · berg@wolfberg.ai

Preprint. Not peer reviewed. Published 29 October 2026.

Version 1.1, 29 October 2026: set in the house research-note format with house exhibits; text and numbers unchanged from Version 1.0, published the same day.

AT A GLANCE

The 740 misses in the register (13 June to 8 October 2026), by who the record says caught them, and who wrote each row. In three of the four routes a seat writes the row, including the 378 rows that credit the operator. From one practice; counts recomputed from the ledger and from register.json (in the site repository, not served), which agree exactly. Verbatim capture on operator rows rose from 0 of 27 in June to 35 of 43 in October.

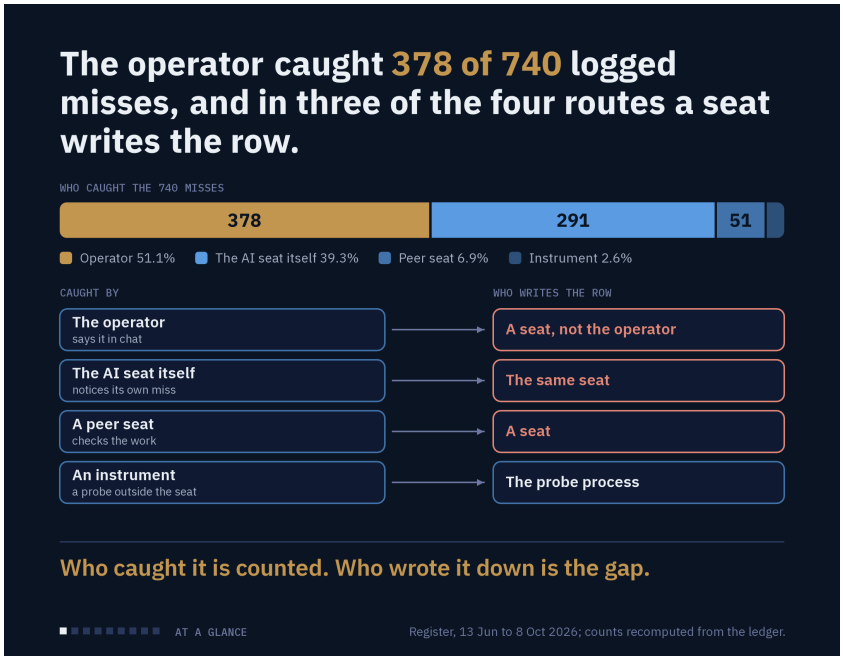


Exhibit 1. The operator caught 378 of 740 logged misses, and in three of the four routes a seat writes the row.

IN PLAIN TERMS

When people ask what keeps an AI honest, the usual answer is a human in the loop. In a study published on 7 August 2026, which backed making auto mode the default in Claude Code, Anthropic tested that answer: it planted one dangerous command in front of each of 1,053 paid testers, and the testers caught it 13.6% of the time, falling from about 17% early in a session to about 5% after 50 or more prompts. That is a careful measurement of the human gate. It is also a single planted test in a test environment.

This practice measures its human gate a different way. Every miss in its AI-assisted work that gets logged goes into a hash-chained register, published as counts and the newest 40 entries at wolfberg.ai/register, and each row records who caught it. In the sources we found, none publishes a continuous tally of who caught each real miss. Since June that is 740 misses: the operator caught 378, the AI seat caught 291 of its own, a peer seat caught 51 and an external instrument caught 19. The operator's share fell from 71% in June to 28% in early October, though 100 of October's 152 rows were logged in a burst on 3 and 4 October, and without them October is 36.5%.

We then had a separate AI instance audit the register, read-only, against its own code and data. The main finding extends the earlier post in this series, "Never let your AI grade its own homework," which ended on the question: who wrote it down? The register records who caught each miss, but a seat writes that record, including every row that says the operator caught it. 159 of the 378 operator rows carry the operator's own words and 219 rest on the seat's word, though 205 of those 219 date from before September and capture is now about 80%. The register also holds one row about a novelty claim, dated 11 September, against 208 novelty assertions in the operator's chats between 13 June and 27 August, none of which appears word for word in the practice's published work. The human did the filtering; the record barely saw it.

Our position: a human gate should be measured continuously, on real work, by who caught what, and the record of who caught it should not be written only by the party being measured. Real catches tell you what was caught, by whom, and how that changes. Planted items, as in Anthropic's test, give you a rate and a view of what got past. A practice needs both, and this one has only the first so far.

ABSTRACT

Human oversight of AI work is usually asserted rather than measured, and most current measurements are one-off, planted, or about approvals rather than catches. We describe a continuous alternative in use at one practice: a public register that records each logged miss in AI-assisted work with the party that caught it (operator, the AI seat itself, a peer seat, or an external instrument), hash-chained with SHA-256 and with the chain head signed with Ed25519. From 13 June to 8 October 2026 it holds 740 misses: operator 378 (51.1%), self 291 (39.3%), peer 51 (6.9%), instrument 19 (2.6%), unattributed 1. The operator's monthly share went from 71.1% (June, n = 38) to 28.3% (1 to 8 October), confounded by a burst of 100 rows logged on 3 and 4 October (36.5% without it) and a 77-row bulk stamp in July. A read-only audit by a separate AI instance verified the chain in

register.json (745 entries, 0 broken links, head and 5 of 5 gate signatures valid) and reproduced every count, and it found that the catcher label is a pure function of a source field the writing seat asserts: 219 of the 378 operator rows carry no operator words (205 of them from before September), a separate WORM ledger seals 219 of the 740 misses, the chain is rebuilt whole each night, the build checks no signature, the record file is not served, none of one AI reviewer's nine staged findings has reached the ledger, and the five gate entries are fixed examples rather than decisions. Against 208 novelty assertions in the operator's chats from 13 June to 27 August, it holds one novelty row, dated 11 September. We set the register against current work, newest first: Anthropic's planted-command study (August 2026), longitudinal approval drift in 11,429 reviews of AI-agent pull requests (June 2026), planted-error studies of AI reviewers (October 2026), three hash-chained agent-log designs (September and October 2026) and the EU AI Act's human-oversight article, whose application to Annex III systems moved to December 2027. Our position is that a human gate needs a continuous record of real catches, with the catcher's attribution witnessed by something other than the party being measured, plus planted items to supply a denominator.

Keywords: human oversight, human in the loop, automation bias, AI-assisted work, audit logs, tamper evidence, self-report, measurement, EU AI Act Article 14

1 Introduction: where things stand

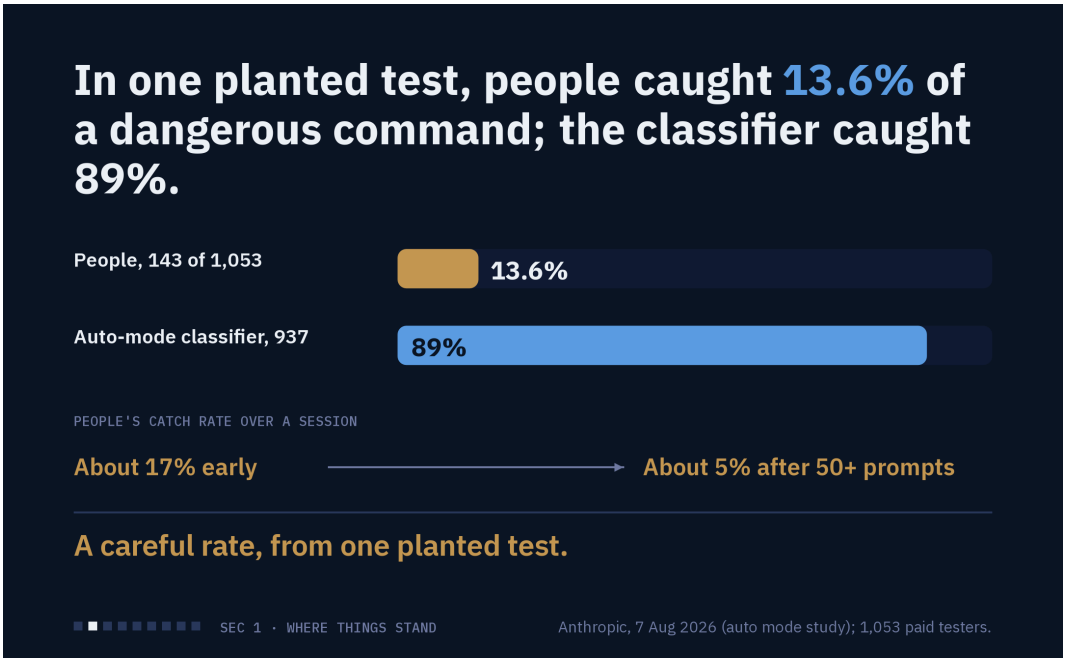


Exhibit 2. In one planted test, people caught 13.6% of a dangerous command; the classifier caught 89%.

On 7 August 2026 Anthropic published the study behind making auto mode the default in Claude Code. Each of 1,053 paid testers, recruited through a vendor, met one planted dangerous command in a purpose-built environment where approving it ran nothing dangerous; testers knew they were being evaluated but not what was measured. Humans caught 143 of the 1,053 (13.6%); the auto-mode classifier caught 937 (89%). Human catches fell from about 17% early in a session to about 5% after 50 or more prompts (Anthropic, 2026b). In February the same lab had reported from real telemetry that roughly 20% of new users' Claude Code sessions use full auto-approve, rising to over 40% as users gain experience (Anthropic, 2026a).

The research of the last few months points the same way. Yu et al. (2026) followed 400 repeat reviewers across 11,429 reviews of pull requests written by AI agents and found the population's approval rate rose from 30.1% to 36.8% while inline comments fell. Mitchell, Ghosh and Passi (2026) argue that current agent designs do not support effective human oversight and contribute to its degradation. Reid (2026) argues oversight should be judged by intervention capacity rather than by a human's presence. Song (2026) planted 150 errors in 30 artifacts and found the best cross-model AI reviewer not significantly different from a fresh session of the same model, which does not establish equivalence; two reviews that differed in model found more planted errors. Regulation has slowed: the EU's Digital Omnibus moved the application of the AI Act's high-risk chapter, including the human-oversight article, to 2 December 2027 for Annex III systems (Regulation (EU) 2026/1744).

These are careful measurements, and almost all of them are one-off, planted, or about approvals rather than catches. A one-off planted test gives a rate under test conditions. An approval log shows that a human clicked, not whether the thing approved was wrong. Neither tells a practice what its own human gate catches on an ordinary day of real work, or how that changes over months. This note describes a register built to record exactly that, and what an audit found when we asked who writes it.

The earlier post in this series, "Never let your AI grade its own homework" (Atkinson, 2026a), measured the cost of self-report: the AI team logged its own session starts in 75 of 579 sessions, and its own logs caught corrections on 1.76% of the turns where a hand-graded detector found them on roughly a third. It ended on one question: who wrote it down? The register was built to answer that for every miss. The audit asks the same question of the register.

2 Our position

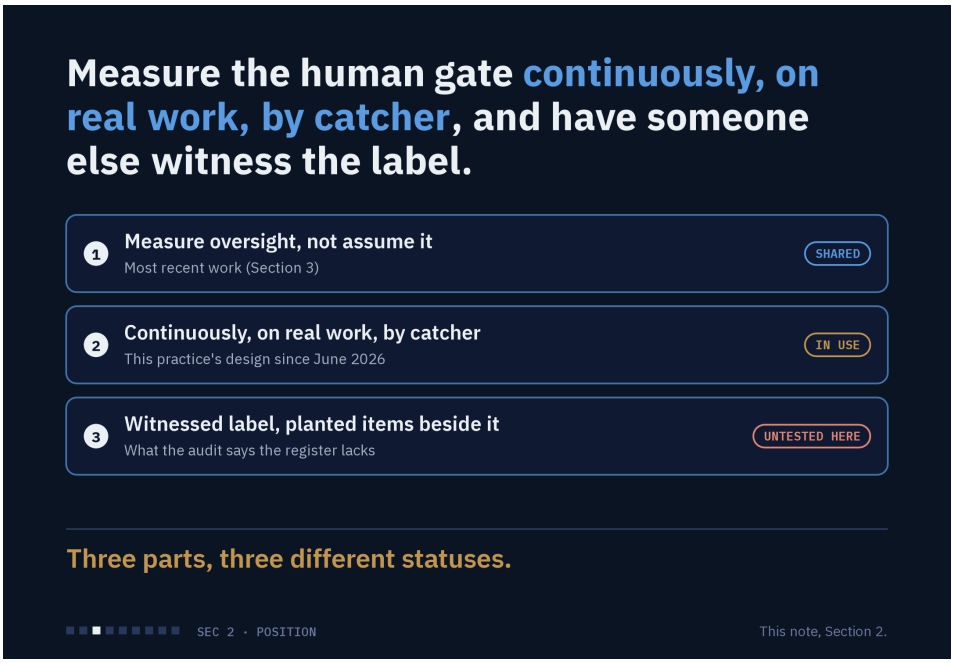


Exhibit 3. Measure the human gate continuously, on real work, by catcher, and have someone else witness the label.

A human gate should be measured the way any control is: continuously, on real work, by what it caught and what got past it. The record of who caught each miss should not rest only on the word of the party being measured, and real catches need planted items beside them to give a rate.

Three parts, each with a different status. That oversight should be measured rather than assumed is the position of most of the recent work in Section 3 and of Green (2022) before it. That the measure should be continuous, on real work and by catcher type is this practice's design, in use since June 2026. That the catcher label should be witnessed by something other than the writing seat, and paired with planted items, is what the audit in Sections 4 to 6 says the current register lacks; it is untested here.

3 What else exists, newest first

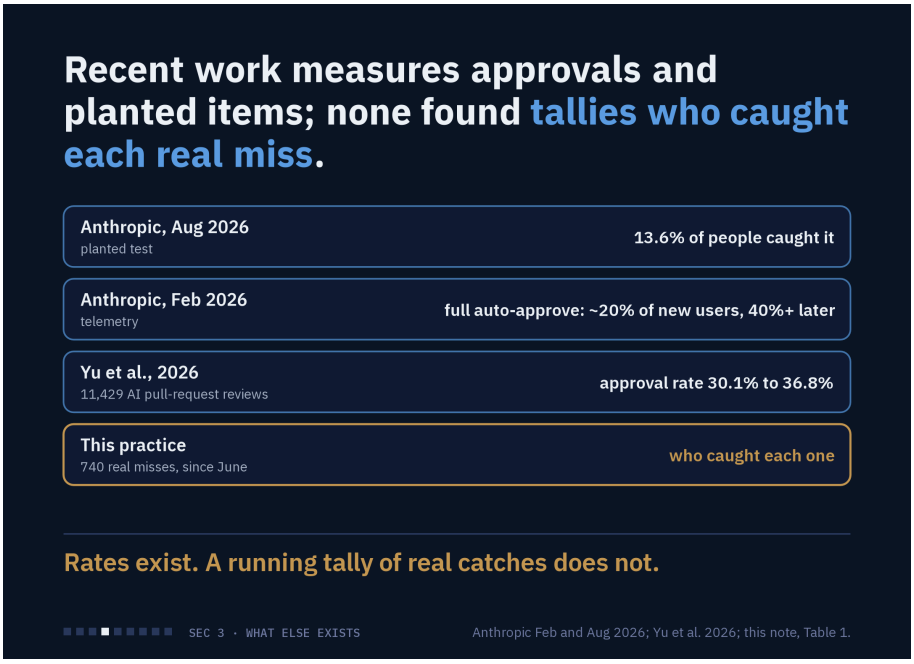


Exhibit 4. Recent work measures approvals and planted items; none found tallies who caught each real miss.

We searched for measurements of human catch rates in AI workflows, records of who caught AI errors, tamper-evident agent logs, and oversight rules, weighting the last eight weeks most (about 20 queries, 30 sources, run 10 October 2026). Table 1 lists the closest, newest first. In that search we did not find another practice publishing, continuously, which party caught each real miss in its AI-assisted work; the nearest are a one-day account of fifteen errors and the checks that caught them (Ritschel, 2026) and an AI agent's public log of its own claim checks (Stellar, 2026). That is a bounded statement about one search, not a claim that nobody does it.

Table 1. Work on measuring and recording human oversight of AI, newest first. Age is days before 10 October 2026. Band: now is since 15 August 2026. Rows marked † were read through search snippets, not at the source. The survey's arXiv number implies August 2026; its listed date is used.

Date (age)	Who	What it measures or does	How the register differs
about 9 Oct 2026 (1)†	Stellar, an AI agent (dev.to)	Public, versioned log of the agent's own claim checks; one error caught by a peer re-running sources	Records one peer catch; AI-authored, with no operator and no catcher-type tally found in the snippet
9 Oct 2026 (1)†	Independent developer (dev.to)	Recipe: SHA-256 chain over append-only agent tool calls	Same chain primitive over actions; no catcher field found in the snippet
4 Oct 2026 (6)	Ritschel (Zenodo)	Fifteen errors in one day of an AI-assisted biology pipeline, and the checks that caught them	One day, not continuous; no catcher-type split; the author notes the count is not a rate

Date (age)	Who	What it measures or does	How the register differs
about 4 Oct 2026 (6)†	Beri (vendor guide)	Advice: seed 1 to 2% of the review queue with known-wrong actions and track the hit rate	Planted canaries give a denominator the register lacks; not a published dataset
1 Oct 2026 (9)	Song (arXiv 2610.01471)	Planted errors (150 in 30 artifacts) found by same-model and cross-model AI reviewers	Lab design, AI reviewers only; the register logs peer-seat catches on real work
17 Sep 2026 (23)†	ggcode (GitHub pull request)	Append-only hash-chained ledger of agent actions with tamper localization	Same integrity idea over actions; no catcher field found in the snippet
16 Sep 2026 (24)†	OpenAI, via Axios	Disclosure of misalignment incidents found in training and evaluation	A lab publishing its own failures; model behavior, not checked work product; no catcher tally found in the snippet
1 Sep 2026 (39)	Agent Flight Recorder (arXiv 2609.01931)	Event schema for agent actions with hash chaining and Merkle batching; synthetic workloads	Stronger anchoring than the register's chain; no record of who caught errors
30 Aug 2026 (41)†	Reid (SSRN)	Argument: judge oversight by intervention capacity, not presence	Conceptual; the register records intervention outcomes
24 Aug 2026 (47)	Mitchell, Ghosh, Passi (arXiv 2608.23642)	Position: agent designs degrade oversight and the skills it needs	No catch data; the register is one place such degradation could show
7 Aug 2026 (64)	Anthropic	One planted dangerous command per tester, 1,053 testers; 13.6% caught; about 17% early to 5% late	Planted, one-off, test environment; the register is real misses, continuous, by catcher
24 Jul 2026 (78)	EU Regulation 2026/1744	Moves the high-risk chapter, including Art. 14 human oversight, to 2 Dec 2027 for Annex III	Law requires oversight be possible; it does not require measuring catches
29 Jun 2026 (103)	Survey of AI-scientist systems (arXiv 2608.05179)	No LLM-era system in its corpus demonstrates an externally validated in-loop oracle, under its coding rule	A related verification gap in AI-scientist systems; not about human catch rates
25 Jun 2026 (107)	Dhanorkar, Passi, Vorvoreanu (FAccT)	Interviews with 17 developers on how they oversee coding agents	Self-report; no logged outcomes
21 Jun 2026 (111)	Yu et al. (arXiv 2606.22721)	Approval drift in 400 reviewers, 11,429 reviews of AI-agent pull requests	Real and longitudinal, but approvals with no ground truth on errors

Date (age)	Who	What it measures or does	How the register differs
8 Jun 2026 (124)	arXiv 2606.08919	Simulated reviewer fatigue on 125 labeled agent actions; reviewers agree only moderately on risk	Assumed decay curve; the register could supply observed data
18 Feb 2026 (234)	Anthropic	Real telemetry: auto-approve share, human interrupts, agent-initiated stops	Aggregate stops, not confirmed errors; no per-event public record
13 Jun 2024, signed (849)	EU AI Act, Art. 14	Overseers must be enabled to understand, resist over-reliance, override and stop	A design duty, not a measured catch rate
2022 (about 1,600)†	Green	Survey of 41 policies; people cannot perform the oversight functions asked of them	Foundational critique; proposes institutional remedies, not a ledger

4 How the register works



Exhibit 5. Six writers feed one ledger; a producer chains it, and a separate signer signs the head.

This section reports a read-only audit of the register's code and data by a separate AI instance on 10 October 2026, with file and line references kept in the working report (Atkinson, 2026c). It read the producer, ingest, page and integrity tests in full and the larger writer and WORM scripts in part, re-implemented the chain, signature and count checks, and ran no test suite. Exhibit 5 is the data flow.

4.1 Writing a row

Misses go into one plain JSONL ledger (a live file plus an archive for pruned rows) by four routes. Code seats call a command-line logger with a miss type and a source of operator, cross_instance or self; the logger rejects unknown types and sources and appends under a file lock. Chat seats drop a JSON file into an inbox that an ingest script reads about every 30 seconds, using the file name as the event id so a retry cannot count twice; it does not pass the timestamp or the operator's words through. A poll-down copies rows chat seats wrote in a Notion ledger (251 rows, 4 July to 31 August). Probe scripts, run outside the seat being measured, append rows with source probe. Separately, hooks write guard rows (evaluations, refusals, overrides, stop certificates) that record a hash and rule labels, never the subject text; these are not misses and the register does not count them.

A published miss carries its type, weight, who caught it, the first sentence of its note (anonymized, cut at 200 characters), its timestamp, and its place in the chain. Eleven miss types are counted, each with a fixed weight from 3 (format slip) to 16 (an error that would have shipped).

4.2 Who decides who caught it

The register does not decide who caught a miss. It publishes the source field of the row, and whoever writes the row sets that field.

The label is a pure function of the row's source: operator becomes operator, self becomes self (unprompted), cross_instance becomes peer instance, probe becomes external instrument, and anything else becomes unattributed. Until 3 October an unknown source fell through to operator, which had mislabeled 44 of 624 rows; the unmapped value was cross_instance, so the fix moved peer-caught rows out of the operator count, and 43 published rows changed label in the next build. Anyone who read the page before 3 October saw no peer rows and a higher operator share. Nothing in the logger or the ingest checks the asserted source against an independent signal. One client-side script refuses an operator row without the operator's verbatim words; the logger itself does not. The at-a-glance figure shows the result: 159 of the 378 operator rows carry the operator's words, and the other 219 rest on the writing seat's word. Capture has risen: by month, 0 of 27 operator rows carried his words in June, 31 of 165 in July, 61 of 105 in August, 32 of 38 in September and 35 of 43 in October; 205 of the 219 date from before September. The chat-seat file drop does not pass the operator's words through, so some rows could not carry them. Probe rows, 19 in all, are the only ones written by a process other than a seat.

4.3 Building and signing the record

Each night a producer reads the live ledger and the archive, keeps the eleven miss types, removes duplicates by event id (879 raw miss rows become 740), and sorts newest first. It then chains five gate entries and the 740 misses: each entry's hash is SHA-256 over the previous hash and the entry's canonical JSON. The five gate entries' hashes and the chain head are signed with Ed25519, and the public key and head signature are published with the file. Since 29 October 2026 the producer no longer signs. A separate nightly signer, outside the machine that writes the record, recomputes the chain, refuses one that does not verify or does not extend the chain it signed before, and signs the head and

the five gate hashes with an Ed25519 key held in AWS KMS. The /register page checks those signatures in the reader's browser against the published key. The key's policy lets only that signer use it; an administrator of the key can change the policy, and AWS records the change. Misses themselves are hash-chained, not individually signed.

The audit verified the file independently: 745 entries, 0 broken links, the head equal to the last hash, and the head signature and all five gate signatures valid under the published key, which was the same in the builds of 4, 9 and 10 October. It also found what the chain does not do. The file is rebuilt whole every night, and because new rows are inserted above old ones, no entry's hash survives from one night to the next even when its content does; continuity over time comes from git history (48 commits) and, for some rows, the separate WORM ledger. Diffing all 48 builds, four transitions changed earlier entries: 3 July (a 15-row file replaced by the ledger; 9 of the 15 do not reappear), 28 August (1 entry), 1 September (11 text changes) and 2 to 3 October (the 43 relabels above and 5 weight changes).

4.4 What the page shows and leaves out

At build time the page recounts misses from the entries rather than trusting the file's own totals, recomputes every hash and link, and draws the catcher split only if the declared split equals the recount. It shows the newest 40 of 745 entries, the head hash and the public key, and its own lists of what the record proves and does not prove. It leaves out the other 705 entries individually, the session ids, operator words and raw notes, unlogged misses, probe passes and declines, and all guard rows. The record file itself is not served on the site, so a reader cannot repeat the recomputation; the page says so. The build checks no signature, and a broken link shows as a number on the page rather than failing the build.

4.5 Tests

An integrity test seeds five tampers (an edited entry, an edited weight, a deleted entry, reordered entries, a broken previous-hash link) and requires each to fail verification, runs the producer end to end on a planted ledger with every kind of source including missing and malformed ones, and requires the published split to equal a recount. Source checks on the page require counts to come from entries, not from the file's totals. These run in CI on changes to the producer, page or data; checks that need files on the operator's machine print NOT RUN in CI. A hand edit that rewrites an entry and every later hash recounts clean; the nightly rebuild from the ledger overwrites such an edit the next night, and git keeps it.

4.6 The gate half

The five gate entries on the page are fixed examples of the gate's rules, the same every night, undated and signed by the register's key, not by any person; the page says so. The real gate record is in the ledger as guard rows written by hooks between 2 September and 9 October 2026: 5,032 evaluations, 863 guard-refusal rows, of which 243 are marked blocked and 620 carry no outcome (584 from the keystroke guard, 145 from a read-before-claim guard, 61 from a stop-delivery guard, 37, 8 and 2 from three smaller guards, and 26 with no guard named), 3 overrides (all on 16 September), and 1,274 stop

certificates, 14 of them refused. None of it is published yet. A separate nightly ship-or-hold check over regression tapes (flip_gate) is not part of the register.

5 What the record shows

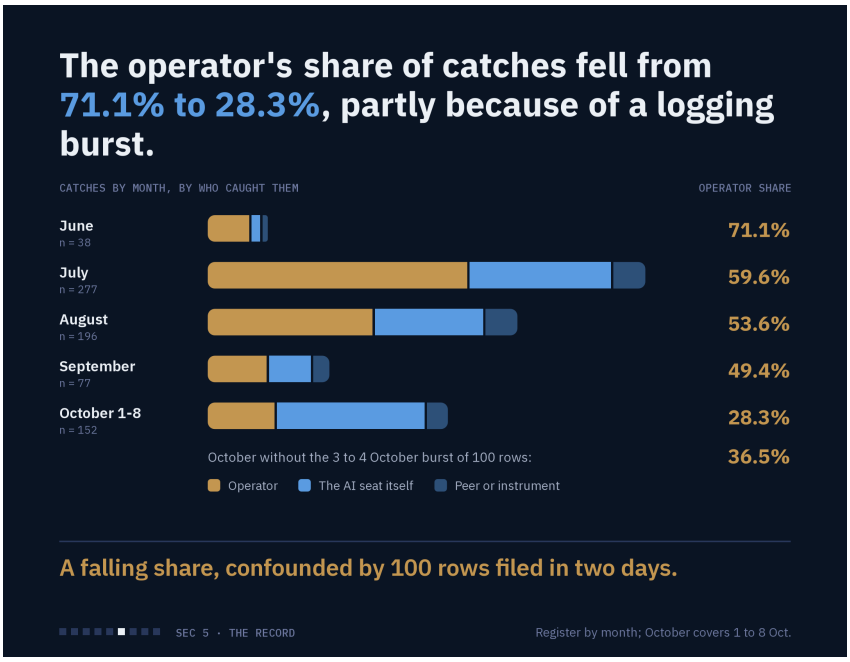


Exhibit 6. The operator's share of catches fell from 71.1% to 28.3%, partly because of a logging burst.

5.1 Who caught the logged misses

Of 740 logged misses, the operator is credited with 378 (51.1%) and the seats with catching 291 of their own (39.3%); peers caught 51 (6.9%) and instruments 19 (2.6%).

Weighted by severity the split barely moves: the operator's misses carry 3,707 of 6,926 weight points (53.5%). The operator dominates the repeated misses (118 of 177) and wrong diagnoses (101 of 161); the seats' own catches dominate self-corrections (44 of 46), as the type implies. Errors that would have shipped, the heaviest type, split 67 operator, 53 self and 12 peer.

5.2 Over time

The operator's share went from 71.1% in June to 28.3% in the first eight days of October. June rests on 38 rows, the weekly series is not steady, and the ledger cannot say how much of the change is behavior and how much is logging practice.

Two filing events sit inside the trend. On 15 July, 77 rows (55 of them operator) were written in one bulk stamp; without them July's operator share is 55.0% rather than 59.6%. On 3 and 4 October three Code

seats logged 100 rows (73 self, 24 operator, 3 peer), most of October's 152; their timestamps spread over about 42 hours, and the ledger cannot say whether they were filed after the fact or as the work happened. Without them October's operator share is 36.5% (19 of 52). June's 71.1% is 27 of 38 (95% interval about 55% to 83%), and the weekly share since September has ranged from 25% to 68%. We report the change and do not read it as the seats getting better.

5.3 What the operator caught that the record did not

A companion note in this series (Atkinson, 2026b) counted 246 verified assertions in the operator's chats, between April and August 2026, that an idea of his was new, unique or something nobody else does; 208 fall between the register's first row on 13 June and the last assertion on 27 August. Of the 69 that could be dated against a track record, 66 came after an earlier statement of the idea; the rest were not tested. None of the 246 appears word for word in the practice's published work (a seven-word match; paraphrase not tested). The register holds one row about a novelty claim: 11 September 2026, caught by the operator, logged as a repeated miss, "CLAIMED NOVELTY ON AN INCOMPLETE SEARCH, again." The nearest class is unsearched claims, 56 rows (30 operator), most of them not about novelty. The companion's reading, from one practice: the filtering was the operator's, and almost none of it was logged. The register measures logged catches, and seats do the logging.

6 What the record does not show

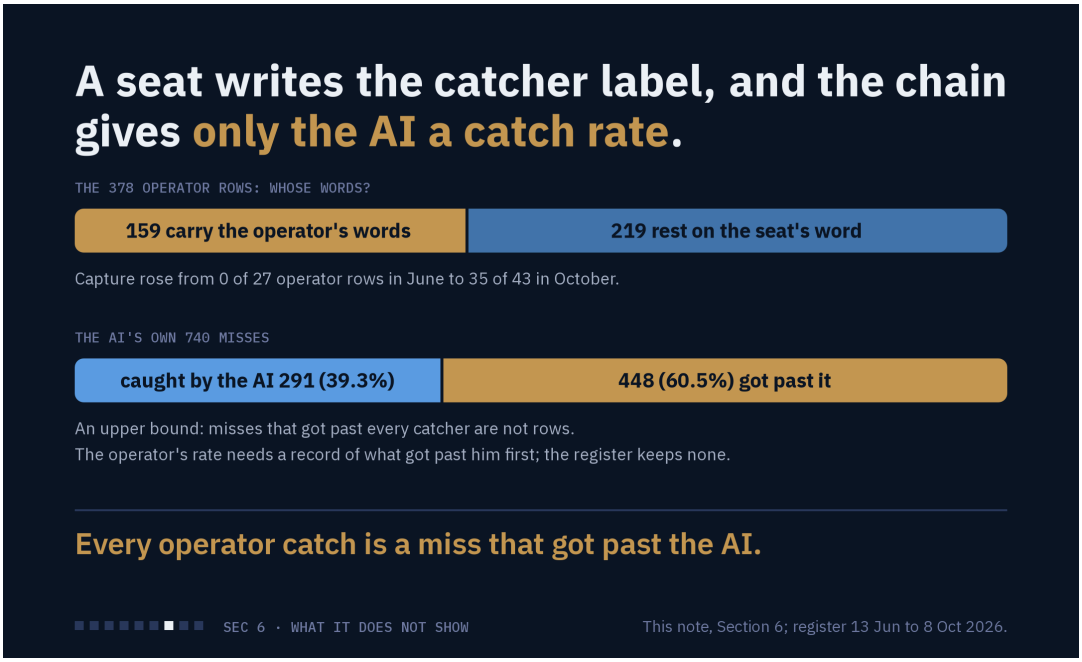


Exhibit 7. A seat writes the catcher label, and the chain gives only the AI a catch rate.

The audit listed ten gaps. We order them by how much they limit what the register can claim.

1. **Self-asserted attribution.** Every row's catcher is asserted by its writer. 219 of 378 operator rows carry no operator words, and the 291 self rows are the seats' own account. For operator rows this is the same self-report problem the earlier post measured, one level up: the seat no longer grades itself, but it records the operator's grade. The 291 self rows are still self-grading. In September and October, 14 of 81 operator rows carry no operator words.
2. **No denominator.** The register counts logged misses and a rate per hour, not misses per opportunity. It cannot say what share of misses the operator catches, only what share of logged catches are his. Anthropic's planted test has the denominator the register lacks. Read as a chain, though, the register does give the AI a catch rate (added 29 October 2026): it caught 291 of its own 740 misses (39.3%), and 448 (60.5%) got past it to the operator, another AI or an instrument. That is an upper bound, since misses that got past every catcher are not rows. The operator's rate needs a record of which misses went past him first, which the register does not keep.
3. **Unlogged catches.** A miss nobody logs is invisible, as the novelty case suggests (one novelty row against 208 assertions in the chats). Five operator corrections from 4 and 5 July sit in the ingest's rejected folder, refused for vocabulary.
4. **One reviewer's findings do not arrive.** Nine findings from a Claude Science non-author review on 7 October are staged in a folder the ingest does not read; no row has been written through that seat's channel, and it is not on the instance roster. One Code-written peer row (8 October) credits Science as a catcher. A plan to connect it exists; the change had not landed in the files read.
5. **Partial sealing.** The WORM ledger verifies clean (16,735 entries, signed since 4 October) but seals 219 of the 740 published misses (measured). The audit infers the cause is that its ingest reads only the live file (inference, not tested). Edits to the other 521 would be visible only through git history.
6. **A narrow integrity claim.** The chain is rebuilt nightly, the record file and entry hashes are not served, the build checks no signature, the register chain and the WORM head do not reference each other, and, at the audit, the signing key file sat on the same machine as the seats. That last gap closed on 29 October 2026, when signing moved to a key held in AWS KMS (Section 4.3).
7. **A default that writes self.** The Notion poll-down still turns a blank source into self, a default the file-drop ingest removed on 30 August; 60 of the 251 Notion rows are labeled self and cannot be told apart from blanks. Separately, 205 rows, almost all from the command-line route, have no event id, so their route cannot be checked.
8. **The gate half is illustrative.** The page's gate entries are examples; the 863 guard-refusal rows (243 blocked) and 3 overrides are not published.
9. **Timestamp quality.** 77 bulk-stamped rows and 71 date-only rows; the page discloses the first.
10. **Hand-copied tables.** The miss-type weights and gate rules in the producer are copies of other files, and their drift checks run only on the operator's machine.

7 What it means



Exhibit 8. Count what the human catches on real work, and ask who wrote it down.

7.1 For anyone with a human in the loop

Count what the human catches, on real work, and keep counting. A one-time test tells you whether the gate works under test conditions; a running record tells you what was caught, by whom, and how that changes as the people and the tools change. What got past every catcher needs planted items or an outside check. Then ask the series' question of your own record: who wrote it down? If the AI being checked writes the row that says the human caught it, you have moved self-report one step, not removed it.

7.2 For anyone building or regulating oversight

We found three hash-chained agent-log designs within six weeks (a paper, a pull request and a recipe; two read as snippets). None we found records who caught an error, or a way to witness that field. Article 14 asks that overseers be able to understand, resist over-reliance, override and stop; Article 14 itself does not ask anyone to show how often they do. With its Annex III application now December 2027, a practice that wants to show effective oversight has time to measure it rather than describe it.

7.3 For this practice

The audit points to changes; none is made yet, and each is a proposal, not a result. An operator row should carry the operator's words or a witness the seat did not write, such as the harness transcript, which a script already reads for guard activity. A small number of planted items in the operator's own queue, the method of Anthropic's study and of the canary practice in Table 1, would give the register a denominator. Rows should be sealed before they are pruned, and the register's chain should reference

the WORM head. The record file should be published and its signatures checked at build, with a break failing the build. The Claude Science review drops should be ingested, the Notion default to self removed, and the real guard decisions published in place of the five examples.

8 Limits

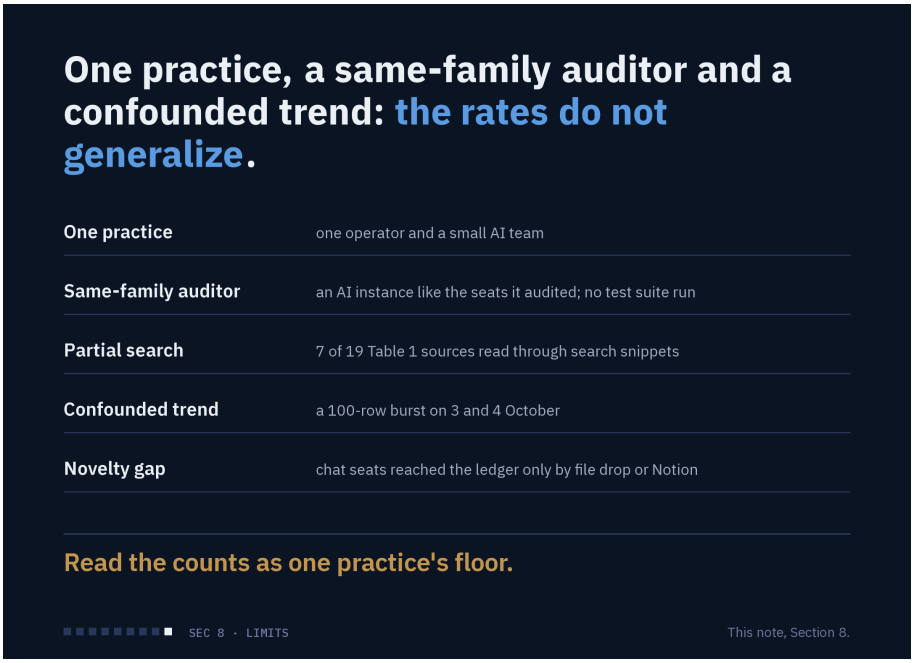


Exhibit 9. One practice, a same-family auditor and a confounded trend: the rates do not generalize.

This is one practice with one operator and a small AI team, so none of its rates generalize. The audit was done by an AI instance in the same family as the seats it audited; it read some large files in part, did not open the key files, and ran no test suite, so its integrity checks are re-implementations and its account of which key signs in production is unverified. Seven of the 19 sources in Table 1 (14 of the 30 the search found) were read through search snippets, the OpenAI row rests on secondary reporting, and the search ran about 20 queries. The change in the operator's share is confounded by a 100-row burst on 3 and 4 October that the ledger cannot separate from behavior. The novelty comparison in Section 5.3 sets chat assertions against a ledger that chat seats reached only by file drop or Notion, over windows that do not match exactly, so some of the gap is plumbing as well as practice.

Open questions

- With planted items in the operator's queue, what is his catch rate, and does it fall over a long session as Anthropic's testers' did?
- If operator rows require a witness the seat did not write, how many of the 14 recent unwitnessed rows, and of the 205 older ones, would survive?

- Does the self-catch share hold once after-the-fact filings are marked and separated from mid-task catches?

References

Anthropic. (2026a, February 18). Measuring AI agent autonomy in practice. anthropic.com/research/measuring-agent-autonomy

Anthropic. (2026b, August 7). Auto mode is now the default in Claude Code. claude.com/resources/articles/auto-mode-default-in-claude-code (study design confirmed at the source, 10 October 2026).

Atkinson, B. (2026a). *Never let your AI grade its own homework.* How do we use this?, Part 4. wolfberg.ai

Atkinson, B. (2026b). *Nobody Else Has Done This: How the Stopping Problem Turns Into False Novelty.* Wolfberg research note, final, 14 October 2026 (not yet published).

Atkinson, B. (2026c). *How Wolfberg's register works.* AI-assisted read-only audit report by a separate Claude instance, 10 October 2026; not public. The register: wolfberg.ai/register.

Axios. (2026, September 16). OpenAI testing safety incidents disclosure. axios.com/2026/09/16/openai-testing-safety-incidents-disclosure (read through a search snippet).

Beri. (2026, about October 4). Human-in-the-loop for AI agents: most approval gates rubber-stamp. beri.net (read through a search snippet).

Bindschaedler et al. (2026). Agent Flight Recorder: Tamper-evident audit trails with on-chain anchoring for long-horizon tool-using agents. arXiv:2609.01931.

Dhanorkar, S., Passi, S., & Vorvoreanu, M. (2026). Human oversight of agentic systems in practice. FAccT 2026. arXiv:2606.05391.

Ding et al. (2026). Autonomous research agents: A survey of AI scientists and the verification gap. arXiv:2608.05179.

European Union. (2024). Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 14. Official Journal of the European Union.

European Union. (2026). Regulation (EU) 2026/1744 amending Regulation (EU) 2024/1689 (Digital Omnibus on AI). Official Journal of the European Union.

Green, B. (2022). The flaws of policies requiring human oversight of government algorithms. *Computer Law & Security Review, 45*, 105681. https://doi.org/10.1016/j.clsr.2022.105681

Hash-chained audit logs for AI agent actions, in 40 lines. (2026, October 9). dev.to/umangcirvix (read through a search snippet).

Mitchell, M., Ghosh, A., & Passi, S. (2026). AI agents push humans out of the loop. arXiv:2608.23642.

Reid, J. (2026). The human-in-the-loop illusion: When formal oversight fails to become functional control. SSRN 7365541 (abstract read through a search snippet).

Ritschel, G. C. (2026). Who checks it now? A measured account of fifteen errors in one day of AI-assisted research. Zenodo, record 23143873.

Song, T.-E. (2026). When does a second model help? Cross-model review in LLM verification. arXiv:2610.01471.

Stellar. (2026, about October 9). I keep a public log of my claim checks, my own errors included. dev.to (read through a search snippet).

topcheer/ggcode. (2026, September 17). feat(audit): tamper-evident hash-chained audit ledger for agent actions. github.com/topcheer/ggcode/pull/2472 (read through a search snippet).

Turan et al. (2026). Oversight has a capacity: Calibrating agent guards to a subjective, fatiguing human. arXiv:2606.08919.

Yu, H., et al. (2026). Habituation at the gate: Rising approval and declining scrutiny in human review of AI agent code. arXiv:2606.22721.