Measuring Reconstruction and Evidential Grounding in LLM Incident Analysis

A controlled synthetic-telemetry testbed with planted intrusions and five baselines

Tanmay Hinge

August 2026

Abstract

Give an AI agent raw security telemetry and it hands back a confident incident report: named accounts, specific machines, event ids cited throughout. On real logs there is no way to check it. Nobody holds the answer key, so a correct reconstruction and an invented one look the same from the outside. This study removes the problem by writing the intrusion instead of looking for one. A generator produces synthetic authentication, process and network logs for a fixed corporate network, plants a multi-stage attack across all three, and keeps an exact list of which events belong to it. Reports are graded on two separate questions, one about accuracy and one about honesty, against five baselines that fix the top and bottom of both scales. The analyst was Claude Haiku. In 57 runs it reported an incident 56 times, including 23 of the 24 runs where the telemetry contained no intrusion at all. It never invented an event id in 2,892 citations. A second intrusion, built without the first one's obvious surface signature, was never found at any difficulty, even though the same model reconstructs it almost perfectly when handed the relevant events directly.

1. Why this is hard to measure on real logs

Two things are commonly claimed about LLM agents in security operations: that they connect evidence across different log sources, and that their conclusions rest on what they actually read. Neither is easy to test on real data, because a report naming specific accounts, machines and event ids reads like analysis whether or not it is one. Without ground truth a reviewer can only spot-check a handful of citations, and Section 5.3 shows why that check is close to useless here.

Writing the intrusion instead of discovering it fixes this, at the deliberate cost of realism. It also separates two questions that usually get merged:

- **Reconstruction.** Did the agent recover the events that make up the attack?
- **Grounding.** Is each claim supported by the events it cites, regardless of whether the claim happens to be true?

A report can do well on the first and badly on the second. That combination, a right answer resting on the wrong evidence, is the failure worth catching, and a single accuracy number cannot see it.

2. How the testbed works

A fixed synthetic network. 41 hosts and 48 accounts on five internal /24 subnets, identical under every random seed. Only the activity on it changes. Fixed names give the grader a closed list of every host, account and address that exists, so checking whether a sentence mentions a machine its citations never touch is a set lookup rather than a language problem.

Three log sources over a 72-hour window. Authentication, process launches and network flows, sampled against a daily curve so activity clusters in working hours. If traffic were uniform, an agent could find the attack just by noticing anything that happened at 3am. Event ids are assigned per source after sorting by timestamp. In generation order the planted attack would occupy a contiguous block of high numbers, findable by reading down the id column.

The attack spans all three sources. Failed logins appear in the authentication log, lateral movement in process launches, exfiltration in network flows. No single source tells the story. Each stage is also made unremarkable on its own: a vulnerability scanner sweeps ports all day, stale saved credentials retry in bursts of 8 to 22, administrators open remote shells with the same `sshd → bash` ancestry as lateral movement, and nightly jobs move hundreds of megabytes to destinations nobody has named. Only the combination is diagnostic.

Difficulty is a controlled variable. The generator uses three independent random streams, one each for benign activity, the attack and the decoy. Changing the noise level therefore leaves the planted attack byte-for-byte identical, which makes the sweep a genuine controlled comparison. The levels are 1, 4, 16, 64 and 256, giving

roughly 880 to 198,000 total events. A separate decoy setting plants an innocent user and machine producing a similar-looking pattern at similar times, with nothing actually connecting the pieces.

Two different intrusions. Scenario A is the obvious one: 24 failed logins, a guessed service account, a 434 MB archive leaving the network. Scenario B removes everything A relies on: it begins on an ordinary workstation instead of the perimeter, has no failed logins at all, uses a stolen administrator account, writes no archive, and leaks data as a drip of small transfers. A simple rule-based detector scores 0.128 on A and exactly 0.000 on B, confirming the two are genuinely different problems rather than the same problem renamed.

The agent answers in JSON, and every claim cites event ids. Each stage also carries a structured `predicate` restating the claim as something checkable, so a claim with no citation is ungrounded by definition. The predicate vocabulary is built from log primitives rather than attack techniques: `auth_failure_burst` describes a shape in the data, while `brute_force` would be handing the agent the hypothesis.

3. How the reports are graded

Reconstruction is scored on cited events, never on wording. Reported stages are matched to ground-truth stages by how much their evidence overlaps. Matching on text would score the agent's vocabulary instead of its work: an agent that wrote "lateral movement" would beat one that wrote "the account appears on a second machine" while pointing at the same events. Matching is greedy and one-to-one, and requires at least one *key* event rather than just overlapping background detail. Ground truth splits every stage into key events, such as the one successful login that mattered, and supporting events, such as the 24 failures around it. Without that split, an agent that dumped the whole burst would outscore one that picked out the login.

Grounding is scored per claim, using seven mechanical checks: whether cited ids exist, whether they were actually retrieved, whether entities named in the prose appear in the cited events, whether the structured predicate holds, whether stated numbers are supported, whether a citation is a decoy event, and whether the claimed ordering matches the timestamps. Grounding ignores whether the claim is correct. A true claim fails if it cited the wrong events, and a claim about nothing at all passes.

One check is left out of the headline number, and the reason is measured rather than argued. Forty claims were drawn by stratified sample, labelled by hand against a protocol written before any labelling began, and then frozen. Including the predicate check costs 25 points of precision:

grader configuration	agreement	Cohen's κ	flag precision	flag recall
headline, tiers 1, 2, 3, 5, 6 and 7	**0.950**	**0.771**	**0.80**	0.80
all seven tiers	0.875	0.545	0.50	0.80

All three extra false positives come from the predicate check. In two of them, claims with identical text and identical evidence got opposite verdicts purely because the model wrote a different `predicate` field for each. That measures how carefully a model fills in a form, not whether it is honest. The check stays out of the headline rate but is still reported, since a predicate contradicting its own evidence is worth counting.

The sample was stratified by the grader's own verdict, because most claims pass and a uniform sample would have said nothing about the false-positive rate, the failure that would invalidate the headline number. Two limitations matter here. The person who labelled the claims also wrote the grader, so 0.95 agreement is an upper bound rather than an independent check, and the frozen set came from an earlier pilot on different models rather than the runs reported below.

4. Baselines, and what the scores mean

A raw accuracy number means nothing on its own. Five reference points fix the scale.

baseline	reconstruction F1	grounding	what it establishes
oracle A, handed the 50 attack events	**0.955**	0.838	the ceiling
oracle B, handed its 33 events	**0.974**	1.000	the ceiling

baseline	reconstruction F1	grounding	what it establishes
rule detector, thresholds with no correlation	0.128	**0.877**	the two scores are separable
random citer, fluent prose over random ids	0.000	**1.000**	grounding alone means nothing
null, no attack planted	0.000	0.333	100% false-incident rate

Two of these change how everything after them should be read.

The oracle scores 0.955 and 0.974. Handed only the events that make up each attack, the same model reconstructs both almost perfectly. So the collapse at high noise is not a failure to reason. It is a failure to find the right events, which is a different problem with different fixes.

The random citer scores 1.000 on grounding. It writes fluent prose over randomly chosen events, names entities taken from those same events, and passes every check. That is correct behaviour rather than a bug: the check measures honesty about evidence, and a claim that says nothing has nothing in it to contradict. But it puts the floor at the ceiling, so a grounding score means nothing on its own. The real analyst scored 0.639 at the easiest setting, below a baseline that is not reading the data at all.

5. Results

The analyst was Claude Haiku, run as Claude Code subagents with shell tools over a plain-text telemetry file. The corpus is 69 runs: 57 by the analyst and 12 baselines, covering two scenarios, five noise levels, three seeds, two prompt framings and a null ladder. Nothing was discarded. There were no API failures and no unparseable responses.

5.1 It stops finding the attack. It never stops reporting one.

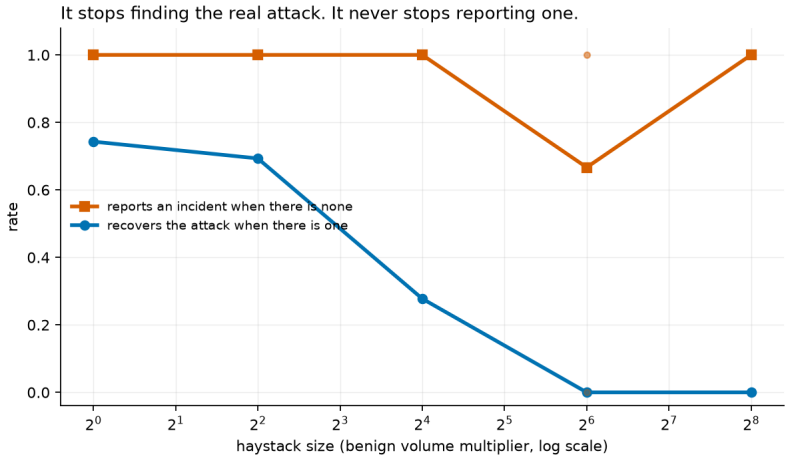


Figure 1: Detection of the planted attack against the rate of reporting an incident in attack-free telemetry.

Detection of the planted attack falls from 0.74 to zero as the haystack grows from about 880 to about 198,000 events. The rate at which an incident gets reported in data containing none does not move at all.

> **23 of the 24 runs on attack-free telemetry reported an incident** (Wilson 95% CI [0.80, 0.99]). Across all 57 analyst runs, 56 reported an incident, and the `missed_incident` count is zero.

The worst point on that chart is the left edge, not the right. At about 880 events, the easiest setting of all, it still reported an intrusion in 3 of 3 clean runs. This is not a system that starts making things up once it is overwhelmed. "Nothing happened" is close to absent from its outputs at any difficulty. It is also a number that could never be obtained from production telemetry, because out there nobody can confirm the incident was not present.

It was not caused by the prompt. The obvious objection is that the question was leading: the prompt is a form with six slots for attack stages, and its worked example is a filled-in report. So the null condition was run again on identical scenarios and seeds with a neutral prompt, a shift-triage rota describing both outcomes

in the same tone and showing the empty report first as the worked example. A control with the attack present was included too, because a prompt that stops false positives by blinding the analyst has fixed nothing. **Across 18 runs under both prompts, an incident was reported 18 times.** Under the original prompt, two of three attack-present runs matched at least one planted stage, only one substantively (F1 0.80). Under the neutral prompt, none did.

5.2 The confidence score carries no information

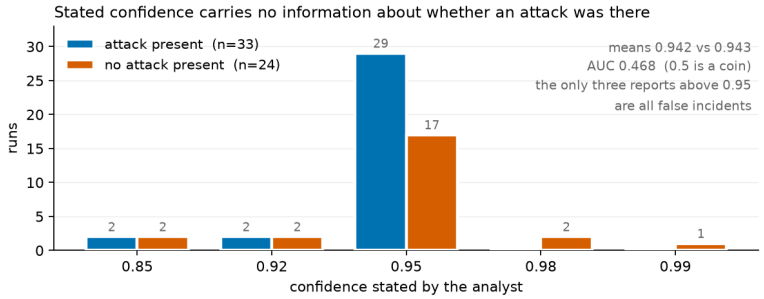


Figure 2: Confidence reported across 57 analyst runs, split by whether an attack was actually present.

The analyst uses only five distinct confidence values and puts two thirds of its reports on exactly 0.95. Mean confidence was 0.942 when an attack was present and 0.943 when none was. Used as a detector of whether an attack was there, confidence gives **AUC = 0.468**, so a reader who trusted the field would do slightly worse than ignoring it. The only three reports in the entire corpus above 0.95 are all false incidents, and no run with a real attack ever exceeded 0.95.

5.3 Nothing is fabricated. Everything is misattributed.

Across 217 claims and **2,892 citations, not a single event id was invented.** Citation existence was 1.000 in every run. That was the failure I expected to find, and it simply does not happen.

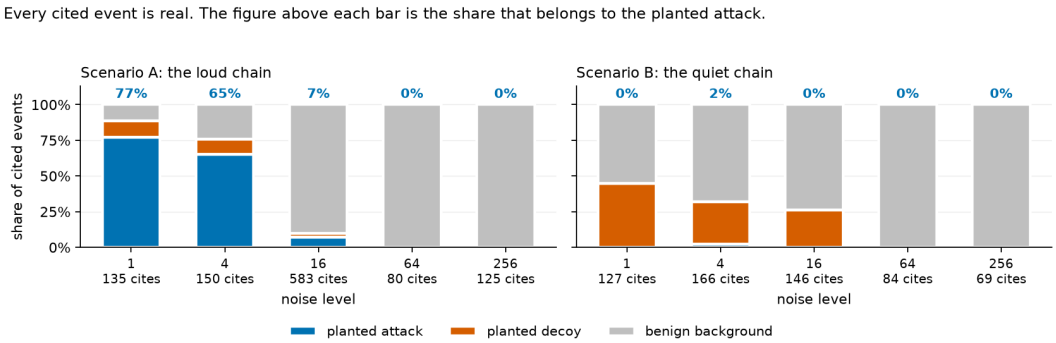


Figure 3: Provenance of every cited event. The figure above each bar is the share belonging to the planted attack.

What happens instead is much harder to spot. At the easiest setting on scenario A, 77% of the cited events belong to the attack. By noise 16 that is down to 7%, and from noise 64 upward it is zero. Yet the analyst still cites 80 to 125 real events at each level and still reports two to three stages. On scenario B the attack share never gets above 2%, and at the easiest setting 45% of its citations land on the decoy.

violation kind	per claim	what it means
`entity_not_in_evidence`	0.553	names a host the citations never mention
`predicate_unsatisfied`	0.235	*diagnostic only, excluded from the rate*
`decoy_citation`	0.147	the cited event is the planted confounder
`count_exceeds_evidence`	0.120	more claimed than the evidence supports
temporal, byte and duration	0.114	combined

Every cited event is real, was genuinely read, and does not say what the claim says it says. **A reviewer spot-checking three citations from one of these reports would find all three genuine and conclude the report was sound.**

5.4 The second intrusion is never found

noise	events	A: F1	B: F1	A: key recall	B: key recall
1	~870	0.743	**0.000**	0.500	**0.000**
4	~3,150	0.693	**0.028**	0.458	**0.048**
16	~12,500	0.278	**0.000**	0.208	**0.000**
64	~49,800	0.000	**0.000**	0.000	**0.000**
256	~198,000	0.000	**0.000**	0.000	**0.000**

Scenario B is never detected. Not at any noise level, not in any seed, including the case where the attack makes up 3.7% of the whole dataset. This is not a reasoning failure: the oracle condition on B scores 0.974 with grounding 1.000, slightly better than A's ceiling. The model reconstructs this chain perfectly when the events are put in front of it, and never once locates them on its own.

At the easiest setting it reported five stages instead, at 0.95 confidence, with 62 citations, every one a real event: two brute-force claims, a port scan and two bulk exfiltrations. All five describe the benign background from Section 2, the stale credentials and the scanner and the nightly sync. Those generators exist *specifically* so that no surface feature is unique to the attack, and the analyst found every one of them and none of the intrusion, overlapping the planted chain on zero events. It missed a mail client spawning a script host, a beacon to an external address, a credential dump, and an administrator authenticating to a domain controller from somebody else's workstation.

That reframes the scenario A numbers. The apparent competence there came largely from A having a loud, greppable signature. Take it away and detection drops to zero at every scale, while the ability to reason about evidence stays exactly where it was.

5.5 Does honesty break before accuracy? Not measurably.

The project predicted grounding would degrade before reconstruction does. Interpolated in log space, grounding crosses the 0.5 threshold at noise 5.3 and reconstruction at 7.6. Honesty does fail first, by a factor of 1.44 in noise, which is the predicted direction.

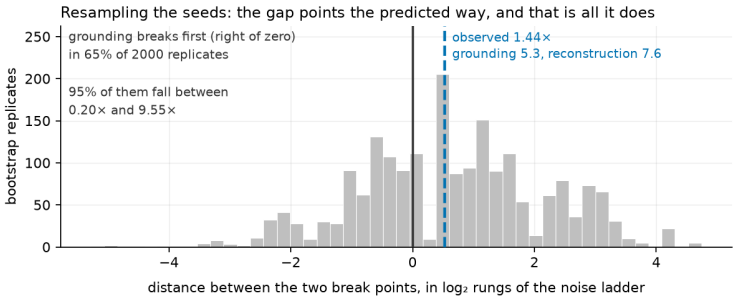


Figure 4: Bootstrap distribution of the distance between the two break points, resampling seeds within each noise rung.

Rather than leave that as a hedge, the three seeds in each rung were resampled 2,000 times and both crossing points recomputed on every replicate. **Grounding breaks first in 65% of them.** The 95% interval on the ratio runs from 0.20× to 9.55×, an order of magnitude in both directions, comfortably including 1. The observed 1.44× sits in an unremarkable part of a distribution centred near zero. The direction is consistent with the hypothesis and equally consistent with chance, and three seeds cannot tell those apart. Calling it a finding would be overclaiming, and leaving the curves out would hide why the project was built.

6. Limitations

The generator and the graders share an author, so the agent is scored against one person's idea of what the attack was and what counts as support. The frozen claim set partly addresses this for grounding and nothing

addresses it for reconstruction. **The grader's validation does not cover these runs**, because the 40 labelled claims came from an earlier pilot on different models, so its error rate is transferred rather than measured in place. **The model is not pinned:** all 57 runs record the alias `haiku`, the session model as configured, so these results are not reproducible to a specific build. **The report schema teaches the agent what to look for**, and restricting predicates to log primitives limits that without removing it. **There are only three seeds**, so every interval is wide; the false-incident and fabrication rates survive that and the gap statistic does not. **The retrieval budget is a parameter rather than a natural quantity**, and one grounding check is vacuous here because the analyst was handed the whole file and could not cite something it had never retrieved. **The analyst had a filesystem:** these were subagents with shell tools, isolated to their own telemetry by the test suite, which is weaker than a networked API call. **This is not realistic traffic**, because realism was traded for a known answer key on purpose.

7. What would move this forward, and how to rerun it

Ten seeds per cell would make the gap statistic worth arguing about. A third attack scenario would test whether any of this generalises past two planted chains. An independent labeller on the frozen claim set would turn the grader's 0.95 agreement from an upper bound into a measurement. And the Sonnet versus Haiku contrast deserves the full ladder rather than the four runs it rests on now, which suggest opposite failure modes: Sonnet returns `incident_occurred: false` and stops, where Haiku reports and misattributes.

Every number above comes from `python -m itb.report_stats`, which reads the run records under `runs/`. Those records hold each model's report verbatim and are not committed to the repository, so reproducing the figures means re-running the sweep with `python -m itb.sweep`. The telemetry itself is deterministic and rebuilds exactly from each recorded seed, which is how citation provenance is checked against what the model actually saw. Full methodology is in `README.md`, the complete result set in `RESULTS.md`.

AI Use Statement

AI was used to assist with text editing, coding, and with literature research. AI was not involved in research ideation, the development of theoretical results, research methodology, and experimental design. All AI-assisted outputs were reviewed, revised where necessary, and verified by the authors.