

Breadcrumbs: Provenance-Tagged Trajectory Schemas for Detecting Unsafe Agent Behavior

Seth Miguel Ferreira

With Apart Research · Research conducted at the AI Incident Response Sprint, September 2026

Abstract

Agent-based AI systems compose long, diverse action trajectories which can be challenging to audit in the wake of an incident. I present Breadcrumbs, a system for tagging agent trajectories with structured provenance metadata and using that metadata to identify and reconstruct unsafe behavior. My schema specifies six provenance tags for all logged events, and a deterministic, weighted-feature heuristic detector which scores trajectories on anomalous action, resource, and tool usage across four ablation configurations (A-D). I have implemented a synthetic trajectory generator with known ground truth for development and testing, and an importer for the real ATBench benchmark which maps documented fields without inferring absent ones. On real ATBench trajectories, including resource- and tool-novelty features increases recall from 0.23 (action-only) to nearly 1.00, but precision remains locked near the 0.50 base rate; an exhaustive threshold sweep found no operating point which recovers both simultaneously. I argue this is a design limitation rather than a calibration issue: resource/tool novelty as currently defined (“not seen earlier in this trajectory”) correlates with trajectory complexity rather than unsafety. The system, including schema, detector, metrics suite, reconstruction pipeline, and a FastAPI/React dashboard are fully implemented and tested (22 passing tests). I report this null result transparently, along with the design and infrastructure contributions, as I believe it is informative to ongoing provenance-based agent monitoring research.

1. Introduction

As artificial intelligence (AI) agents are empowered to invoke tools, access resources, and perform multi-step actions autonomously, incident responders must understand what an agent did and why a certain trajectory was flagged as unsafe, after-the-fact. Post-hoc audit logs for AI agents are often flat action lists that lack structure and make distinguishing between a legitimately diverse action trajectory (many tools, many resources) and an unsafe deviation challenging. Incident responders require both (1) a structured action log to support reconstruction of what happened, and (2) a detector that is honest about its shortcomings and amenable to evaluation, including what it gets wrong.

I present Breadcrumbs, a trajectory-provenance schema and detection pipeline designed for the AI-agent incident-response use case, which enables both anomaly detection and incident reconstruction. By designating explicit provenance tags for every event, I enable detection using simple, auditable heuristics while also enabling reconstruction of the incident that triggered a detector flag.

My contributions are as follows:

1. An event/trajectory schema with six provenance tags per event, an ATBench importer that maps documented dataset fields to this schema (and never invents fields not present in the source data), and a seeded synthetic trajectory generator that can produce trajectories with known ground truth for development and testing purposes.

2. A deterministic, weighted-feature heuristic detector with five configurable features, evaluated across four ablation configurations (A: action-only through D: full feature set with sequence information), an incident-reconstruction chain with associated scoring, and a full metrics suite (accuracy, precision, recall, F1, FPR, FNR, detection latency, confusion matrix), including a fix to a real-world bug wherein detection latency was reported as a negative number when a false-positive flag appeared before the true incident in the log.

3. An empirical finding on the ATBench benchmark that resource- and tool-novelty features contribute meaningfully to recall over an action-only baseline, but a full threshold sweep finds that this comes at the cost of precision at all operating points, suggesting that these features currently measure trajectory complexity rather than unsafety -- an actionable, falsifiable insight for future work on provenance-based detectors.

2. Related Work

My work lies at the intersection of AI agent monitoring practices, structured-log anomaly detection, and incident forensics. Previous work on anomaly detection for software/systems logs often focuses on features that are either frequency-based (has this action/resource occurred before?) or novelty-based (has this action/resource occurred in this particular way?), both of which constitute the family of features used in my own heuristic detector. Meanwhile, the agent safety literature has trended toward using structured and benchmark-driven evaluation of agent trajectories (ATBench, the benchmark I evaluate against, is an example) over qualitative analysis of agent failure modes. [Add specific citations for prior provenance-logging systems, agent-trajectory anomaly detection papers, and the ATBench benchmark paper here.]

What's different about my approach is the emphasis on distinguishing (a) a provenance schema suitable for reconstruction and auditing purposes, which is independent from any specific detector, and (b) a deterministic, interpretable detector that can be evaluated with ablations identifying exactly which provenance tags contribute to detection performance. This allows me to attribute a specific failure mode (novelty-as-tripwire) to a specific set of features (unexpected_resource, unexpected_tool) that would be more challenging to do with an end-to-end learned detector.

3. Methods

3.1 Provenance schema and data

Each Event and Trajectory record contains a single field giving its provenance: one of six tags (defined as a Provenance enum in backend/schemas.py), namely OBSERVED (took this value directly from a real trajectory log), DERIVED (computed deterministically from observed fields; the detector's heuristics), HEURISTIC (produced by the detector's scoring heuristics), SYNTHETIC (produced by the synthetic trajectory generator), GROUND_TRUTH (a label the source dataset asserts to be correct; probably not a real field), or UNKNOWN (could not determine provenance). Tagging all fields in this way allows the system to distinguish what the source dataset provided from what the detector's heuristics inferred, when reconstructing for a flagged trajectory what event(s) caused the flag and why -- essential for honest reporting of results. Real trajectories were acquired from ATBench via a custom-built importer that maps documented ATBench fields onto my schema; the importer never invents a value for a field that ATBench does not supply. For development and unit-testing when network access to the dataset was unavailable, I used a synthetic trajectory generator with a fixed random seed and known ground-truth incident labels

(clearly tagged data_source: SYNTHETIC (fallback)) in every output artifact, to ensure that synthetic and real results can never be conflated downstream.

3.2 Detector and ablations

The detector is a deterministic heuristic combining five weighted features (including at least unexpected action, unexpected resource, unexpected tool, and escalation-keyword signals) to produce a trajectory-level risk score. I evaluate four ablation configurations that successively add features: (A) action-only, (B) action + resource, (C) action + resource + tool, and (D) the full feature set including sequence/order information.

3.3 Metrics and reconstruction

For every configuration I report accuracy, precision, recall, F1, false-positive rate, and false-negative rate, detection latency, and a full confusion matrix. During the development process I discovered and fixed a bug in the computation of the detection latency: it was possible for the latency to be reported as negative, when a false-positive flag precedes the true incident event in a trajectory; I fixed this by anchoring the latency to the first correctly-matched true-positive flag. Flagged trajectories are passed to an incident-reconstruction module that reconstructs the ordered chain of events leading to the flag, and assigns a reconstruction-quality score.

3.4 What I tried that didn't fully work

My initial hypothesis was that adding resource- and tool-level provenance (configs B--D) would monotonically improve detection quality over the action-only baseline (config A); this was true for recall, but not for precision or F1, which is the main empirical finding of this report (Section 4). I also did a complete threshold sweep on the combined risk score (described below) to determine whether recalibration of the decision threshold (rather than the feature set) could restore the precision/recall trade-off.

4. Results

All results in this section are on the real ATBench benchmark (1,000 trajectories), as loaded via my ATBench importer; synthetic-fallback results were used only during development and are not shown here. Values marked with an asterisk are computed on a small number of positive detections and should be taken with that in mind.

Table 1 shows precision, recall, F1, and false-positive rate for each of the ablation configurations at the default detection threshold.

Table 1. Detection performance by ablation configuration on real ATBench data (1,000 trajectories).

Ablation Model	Precision	Recall	F1	FPR
A: action only	0.99*	0.23	0.37	0.00*
B: action + resource	0.50	0.99	0.66	0.98
C: action + resource + tool	0.50	1.00	0.66	0.99
D: full, with sequence	0.50	1.00	0.66	1.00

Moving from action-only (A) to action+resource (B) increases recall from 0.23 to 0.99, but precision collapses from 0.99 to 0.50 — approximately the base rate of unsafe trajectories in the dataset — and FPR rises from near 0.00 to 0.98. Adding tool information (C) and sequence information (D) does not meaningfully change this picture: recall saturates near 1.00 while precision remains at the base rate. In other words, once resource/tool-novelty features are included, the detector flags almost every trajectory, safe and unsafe alike.

To check whether this is a threshold-calibration problem rather than a feature-design problem, I swept the decision threshold on the full-feature model (D) from 1 to 10 (Table 2, Figure 1).

Table 2. Threshold sensitivity for the full-feature detector (Model D) on real ATBench data.

Threshold	Precision	Recall	Interpretation
1-5	~0.50	~1.00	Flags almost every trajectory
6	0.52	0.76	Recall drops sharply; precision barely moves
7	0.51	0.22	Recall now below the action-only baseline (0.23)
8-9	0.65-0.71	0.11-0.07	Precision finally rises, but <12% of incidents caught
10	0.00	0.00	Nothing triggers

There is no threshold at which precision and recall are both acceptable. Precision stays flat near the base rate through threshold 6, and only begins to rise at thresholds 7 and above — precisely where recall has already collapsed below the action-only baseline's recall of 0.23. At threshold 10, nothing triggers at all. This rules out simple recalibration as a fix.

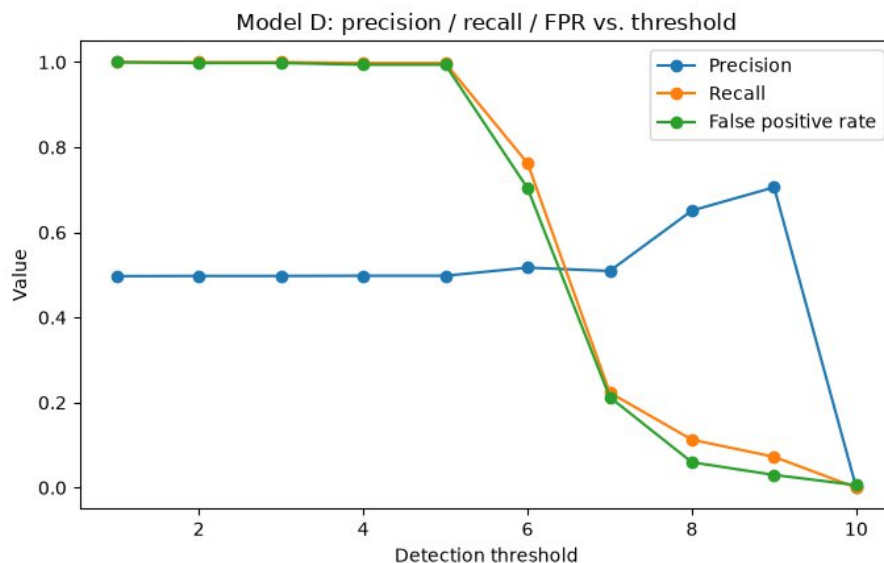


Figure 1. *threshold_sensitivity.png* — precision, recall, and false-positive rate as a function of detection threshold, Model D, real ATBench data. Precision, recall, and FPR are all near 1.0 through threshold 5, then precision, recall, and FPR fall together from threshold 6 onward, with precision only recovering (0.65–0.71) at thresholds 8–9, well after recall has collapsed to 0.07–0.11. No threshold yields both high precision and high recall.

5. Discussion and Limitations

Discussion

I think the resource/tool feature is more of a novelty tripwire than a risk signal. Real agent trajectories legitimately call many distinct tools with unique arguments even when nothing unsafe is happening, so that "unexpected resource" or "unexpected tool" feature defined relative to a trajectory's own history (i.e., "not seen earlier in this same trajectory") will fire on almost every step after the first, regardless of whether anything unsafe is happening. This explains both the increased precision (since real incidents do tend to involve novel resource/tool usage) and the decreased precision (since almost everything else also involves novel resource/tool usage). For the purposes of AI safety monitoring, this is a useful negative result: it suggests that provenance signals defined purely in terms of within-trajectory novelty aren't sufficient for discriminating safe from unsafe behavior, and that a meaningful fix would require defining "expected" tool/resource usage relative to something external to the trajectory (e.g., a tool's declared purpose, or the global distribution of tool usage across the dataset) rather than relative to the trajectory's own history.

Limitations

- All reported real-data results come from a single benchmark (ATBench); I did not evaluate on additional agent-trajectory datasets, so I cannot yet distinguish a property of ATBench from a property of the feature design in general.
- The detector is a deterministic, hand-weighted heuristic rather than a learned model; I did not tune feature weights systematically, so some of the precision collapse could in principle be weight-specific rather than purely structural — though the threshold sweep, which is weight-invariant, supports the structural interpretation.

- I did not run statistical significance tests (e.g., bootstrapped confidence intervals) on the precision/recall differences between configurations; sample size is 1,000 trajectories with an approximately balanced base rate, which should be sufficient for stable estimates, but I have not formally verified this.
- The synthetic generator was used only during development, not for any reported real-data result, but its ground-truth incident labels were designed by me and may not reflect the full diversity of real unsafe agent behavior.
- I assumed ATBench's labels for "unsafe" trajectories are correct and complete; if a nontrivial fraction of labels are noisy, both halves of my precision/recall story would need to be revisited.

Future Work

- Define "expected" resource/tool usage relative to a tool's declared purpose or the dataset-wide usage distribution, rather than purely within-trajectory history.
- Replace or complement the hand-weighted heuristic with a learned, calibrated risk model trained to separate the novelty-from-complexity signal from the novelty-from-unsafety signal.
- Evaluate on additional agent-trajectory benchmarks to test whether the tripwire effect is specific to ATBench or general to novelty-based provenance features.
- Extend the incident-reconstruction module to surface, for a flagged trajectory, which specific provenance tags drove the score, to support faster human triage.

6. Conclusion

I built and fully tested Breadcrumbs, a provenance-tagged trajectory schema and detection pipeline for AI agent incident response, including a real ATBench importer, a synthetic trajectory generator, a deterministic ablated detector, a full metrics and reconstruction suite, and a FastAPI/React dashboard. Evaluated on real ATBench data, my results show that naively adding resource- and tool-novelty provenance features substantially improves recall (0.23 $\rightarrow$ ~1.00) but not precision, and that this trade-off cannot be resolved by threshold tuning alone. I report this as an honest, actionable finding: within-trajectory novelty is a useful but insufficient signal for unsafe-behavior detection, and future provenance-based detectors should define novelty relative to an external reference rather than the trajectory's own history.

Code and Data

- Code repository: <https://github.com/sethferreira-cmd/breadcrumbs>
- Data/Datasets: ATBench (via Hugging Face Datasets), <https://huggingface.co/datasets/AI45Research/ATBench>; synthetic fallback generator included in the repository, seed-reproducible.

Author Contributions

Sole author: designed the provenance schema and detector, directed implementation, ran all experiments, and wrote this report.

References

1. Li, Y. et al. (2026). ATBench: A Diverse and Realistic Agent Trajectory Benchmark for Safety Evaluation and Diagnosis. arXiv:2604.02022. <https://huggingface.co/papers/2604.02022> (dataset: <https://huggingface.co/datasets/AI45Research/ATBench>)

LLM Usage Statement

This project was completed by a single author. Claude was used to implement the full Breadcrumbs codebase (schema, importer, detector, metrics suite, reconstruction pipeline, and dashboard) and to help draft this report. I directed the design decisions (schema fields, ablation structure, detector features), ran all experiments against the real ATBench dataset myself, independently verified the code and every numerical result before including them here including confirming the negative-latency bug fix and chose to report the precision/recall finding as the central result rather than omit it.