INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (ISSN: 2692-5141)
VOL.06 ISSUE 02 (2026)

Research Article

Faogf: Auditable Evidence Provenance For Autonomous Financial Ai Agents On Opentelemetry

Sowmith Javvaji
Independent Researcher, New York, NY, USA
ZENEDO: https://doi.org/10.5281/zenodo.24181994

Received: 01 September 2026
Revised: 09 October 2026
Accepted: 22 October 2026
Published: 16 November 2026

Doi: 10.55640/ijdsml-06-02-16
Page No: 261-281

Copyright: © 2026 Authors retain the copyright of their manuscripts, and all Open Access articles are disseminated under the terms of the Creative Commons Attribution License 4.0 (CC-BY), which licenses unrestricted use, distribution, and reproduction in any medium, provided that the original work is appropriately cited.

Abstract

Autonomous AI agents now read regulatory filings, compute figures and distribute reports inside financial firms, but their telemetry cannot tell an auditor which filing versions a decision rested on, under which authority and policy version an action ran, or whether untrusted content was in the model's context. We present FAOGF (Financial Agent Observability and Governance Framework), an OpenTelemetry-compatible audit layer. Its core architectural contribution is context-bound evidence provenance recorded on two paths: a gateway binds every model call and action to the version-exact, content-hashed evidence actually in the model's context, and writes that binding synchronously to a hash-chained evidence store before exporting standard spans, so audit completeness no longer depends on what the tracing pipeline keeps. We evaluate nine audit queries. In 15 controlled experiments with a scripted agent (200 workflows × 5 seeds, and a real OpenTelemetry Collector on two machines), all nine queries were evaluated: FAOGF answered the seven answerability queries (Q1–Q7) exactly in every configuration, detected 88–97% of spans lost to export-queue overflow (Q8) and 100% of audit attributes stripped by a Collector processor (Q9). Standard OpenTelemetry tracing recovered only 9–12% of protected actions at 10% head sampling and was silently stripped by a common Collector configuration; full prompt logging could not answer authority or policy questions. In a live study on SEC EDGAR filings with gpt-4o-mini and gpt-4o on public Finance Agent Benchmark questions, we scored the two queries the live workload exercises, supporting filings (Q3) and untrusted-context exposure (Q7). FAOGF was exact on both, standard tracing misattributed 4–18% of supporting filings, and full prompt logging used 7–10× more storage. Neither live model followed any of the 78 injected instructions, so enforcement was evaluated on a simulated susceptible agent. There, escalating only the sends requested with untrusted content in context blocked every injected send with 37% fewer human approvals than requiring approval for every send. FAOGF adds about 0.2 ms median overhead per call.

Keywords: Index Terms—AI agents, observability, OpenTelemetry, audit logging, data provenance, financial compliance, prompt injection, tamper-evident logs, LLM governance.

1. INTRODUCTION

Financial firms are moving large language model (LLM) agents from chat assistance to delegated work: an orchestrator agent decomposes an analyst's request, hands subtasks to specialist agents, reads 10-K and 10-Q filings, computes figures and distributes the result. Each of these steps is an action the firm is accountable for. Supervisory and record-keeping obligations in securities regulation, model-risk guidance and the EU AI Act's logging requirement for high-risk systems all presume that the

firm can later reconstruct what a system did and why [12], [13], [14], [15], [16].

The telemetry these agents emit today was designed for debugging, not reconstruction. OpenTelemetry's generative-AI semantic conventions record that a model was called and that a tool ran [1]. They do not record which version of a filing was in the model's context when it answered, which authority the acting agent held, which policy version permitted a report to leave the firm, or whether a web page planted with instructions was visible to the model when it chose a recipient. Worse, the spans that do carry this information travel through a pipeline that is built to drop data: production systems sample traces, bound their export queues and run collector processors that rewrite attributes [2], [3], [4].

Problem statement. The precise gap is this: no existing system produces, for every agent action, a record that is simultaneously (i) *context-exact*—naming the version-exact evidence the model could attend to when it acted, not merely what was retrieved earlier; (ii) *authority- and policy-bound*—naming the delegation and the policy version that permitted the action; (iii) *complete under telemetry loss*—present even when spans are sampled, dropped or rewritten; and (iv) *content-free*—provable without retaining raw financial or client data. Existing tools each cover part of this, and no combination of them covers all four. Distributed tracing and LLM observability capture model and tool calls, but on a lossy path and without context membership. Provenance models such as W3C PROV can express derivations, but no tooling populates them from LLM context windows at runtime. Governance and policy engines decide whether an action is allowed, but do not record which evidence was in front of the model when it asked. Retrieval-augmented and agentic question-answering systems improve what is retrieved and cite sources, but a citation is the system's own claim, not an independently recorded fact. Full prompt logging comes closest on (i), but fails (ii)–(iv). FAOGF is designed to meet all four at once.

We ask whether an audit layer can make these questions answerable exactly, without storing raw financial content, while staying inside the OpenTelemetry ecosystem that operators already run. We study four research questions:

- **RQ1.** Which audit questions can existing GenAI telemetry answer, and how accurately, when agents delegate work and evict content from their context?
- **RQ2.** Does auditability survive the ways real OpenTelemetry pipelines lose or alter data—sampling, export-queue overflow and collector processing?
- **RQ3.** How does an audit layer compare with simply logging every prompt and tool result, in accuracy, storage and retained sensitive content?
- **RQ4.** Can the same audit record serve as a runtime control against prompt injection that targets permitted destinations?

A. Contributions

1. **Context-bound evidence provenance.** FAOGF records, for every model call, the version-exact and content-hashed evidence actually present in the model's context, and links each tool call to the model call that requested it. Ablations show this binding is necessary: without it, supporting-filing precision falls from 1.00 to 0.82 and exposure precision to 0.75–0.86.
2. **Dual-path capture with a fail-closed protocol.** Every governed operation is written synchronously to a hash-chained evidence store before its span is exported; protected actions do not execute without a durable record. In the controlled study, across sampling, queue overflow, collector rewriting and span loss up to 50%, FAOGF answered Q1–Q7 exactly and detected 88–98% of lost spans (Q8), against 0.2–11% for trace-structure heuristics.
3. **An audit-query benchmark (Q1–Q9) and a projection methodology** that evaluates four telemetry designs, two content-logging baselines and six ablations on the same runs, so that every difference is attributable to what was captured.
4. **Evaluation on a real pipeline and real filings.** We run spans through otelcol-contrib v0.163.2 on two machines and run agents on live SEC EDGAR filings with two models; standard tracing misattributed 4–18% of supporting filings on real data, and full prompt logging used 7–10× more storage than FAOGF.
5. **Audit-to-enforcement.** Context binding enables a taint-aware policy. On a simulated susceptible agent (Table XIII), it stopped all injected sends to allowed internal destinations with 37% fewer human approvals than a policy requiring approval for every send. Neither live model followed the injected instructions, so the live study measures exposure (Q7), not enforcement.

6. **A standards-ready vocabulary and auditor tooling:** a Weaver-validated fin.audit.* semantic-convention registry and Merkle-anchored selective disclosure of individual records to auditors.

We build on delegation semantics for agent telemetry proposed by Mishra and Sharad [5] and do not claim delegation tracking as a contribution; our baseline B1 approximates it. The rest of the paper is organized as follows. Section II gives background and Section III reviews related work. Section IV states the problem, threat model and audit queries. Sections V and VI present FAOGF's design and implementation. Sections VII and VIII describe the experiments and results, followed by discussion, limitations, implications for practice and conclusion. Section XIII describes the released artifact.

II. Background

A. OpenTelemetry and the GenAI Semantic Conventions

OpenTelemetry (OTel) is the de facto open standard for distributed tracing. A trace is a tree of spans; each span carries a name, timing, status and key–value attributes, and links can relate spans across traces [1], [3]. Trace context crosses process boundaries through W3C Trace Context headers [6]. The GenAI semantic conventions standardize attributes for model calls (gen_ai.operation.name = chat), agent invocations and tool executions (execute_tool, gen_ai.tool.name, gen_ai.tool.call.id), and make capture of prompts, tool arguments and tool results opt-in because of their size and sensitivity [1]. Attribute vocabularies are published as registries and checked with the OTel Weaver tool [7].

B. Where OTel Pipelines Lose Data

Tracing pipelines are designed to be lossy. Head sampling decides at trace start whether a trace is recorded, typically keeping 1–10% of traffic; tail sampling buffers spans in the Collector and keeps traces that match policies such as errors or specific attributes [2], [4]. The SDK's batch span processor holds spans in a bounded in-memory queue and drops spans when the queue is full. Collector processors then filter, delete, redact or truncate attributes before export. Each mechanism is a reasonable engineering choice for debugging. Each is a silent hole in an audit record.

C. Audit Expectations in Financial Services

Broker-dealers must preserve required records in non-rewriteable, non-erasable form or an equivalent audit-trail alternative [12], and must supervise the activities of associated persons, including technology-mediated ones [13]. Model-risk guidance expects firms to document model inputs, outputs and limitations [14]. FINRA has reminded firms that these obligations apply to generative AI [15], and the EU AI Act requires automatic logging over the lifetime of high-risk systems to support traceability [16]. None of these texts prescribes a schema; together they imply that a firm must be able to reconstruct the evidence, authority and policy behind an automated action after the fact.

D. Delegation in Multi-Agent Systems

In orchestrator–worker designs, the orchestrator issues a narrower grant of authority to a sub-agent, which may run in the same trace, in a linked trace or in an entirely separate asynchronous job. When the hand-off is asynchronous, parent–child structure is lost unless authority is carried explicitly. Mishra and Sharad propose attaching delegation identifiers and lineage to agent spans for this reason [5]; FAOGF adopts their semantics and adds evidence, context and policy binding on top.

III. Related Work

Agent observability. Production LLM observability tools (for example Langfuse, LangSmith, Arize Phoenix and OpenLLMetry) capture prompts, completions and tool calls for debugging and evaluation, increasingly over OTel [8]. Dong et al. survey the data an "AgentOps" platform should collect across an agent's lifecycle [9]. These systems log content; they do not bind decisions to version-exact evidence, record authority or policy versions, or guarantee that a record exists when the tracing pipeline drops data. Our B3 baseline represents this design class at its strongest: every prompt and tool result logged.

Delegation and authority telemetry. Mishra and Sharad attach delegation identifiers, principals and lineage to agent spans so that an action can be traced to the human who authorized it [5]. We adopt this as our B1 baseline. FAOGF's contribution is orthogonal: what the agent saw, which policy version ruled, and how records survive pipeline loss.

Distributed tracing and sampling. Dapper established low-overhead sampled tracing [2]; later work studies which traces to keep, for example by biasing sampling toward unusual executions [4]. Sampling is correct for performance analysis, where a representative subset suffices, and incorrect for audit, where every protected action must be accounted for. FAOGF decouples

the two.

Provenance. W3C PROV models entities, activities and agents and their derivations [10]. FAOGF can be read as a PROV-style graph specialized to LLM context: the novel edge is *in-context-at*, linking a model call to the exact evidence versions it could attend to, which neither PROV tooling nor GenAI tracing records.

Tamper-evident logging. Hash chains make undetected deletion or editing of past log entries computationally hard [17]; history trees and Merkle-based logs add efficient membership proofs and auditor verification [18], [19], as in Certificate Transparency [20]. FAOGF reuses these primitives and contributes their placement in an OTel capture path, including fail-closed behavior and epoch anchoring for selective disclosure to regulators.

Prompt injection and information-flow control. Indirect prompt injection plants instructions in content an agent retrieves [21]. Defenses include detection, prompt isolation and, most relevant here, information-flow control that tracks whether untrusted data can influence privileged actions [22], [23]. Those systems change how the agent is built. FAOGF instead derives taint from audit telemetry at the gateway, so it applies to agents the firm did not write, and we evaluate it against attacks that target destinations an allow-list permits.

Financial question answering and financial LLMs. A large body of work measures whether models answer financial questions correctly. FinQA and ConvFinQA test multi-step numerical reasoning over the tables and text of earnings reports [24], [26]; TAT-QA tests hybrid table–text question answering [25]; DocFinQA extends this to full-length filings [28]; FinanceBench and the Finance Agent Benchmark pose open-book questions over real SEC filings, the latter for tool-using agents [27], [11]. Temporal question answering over semi-structured tables tests whether systems reason correctly about time-indexed facts [35]. Domain-adapted models such as BloombergGPT and FinGPT improve financial language understanding [29], [30]. All of this work scores the *answer*. FAOGF scores the *record*: whether, after the fact, an auditor can establish which evidence version, authority and policy produced an answer, whether it was correct or not. The two are complementary. A system can answer a benchmark question correctly from a filing that is later restated; accuracy benchmarks have no notion of this, while FAOGF's Q4 identifies every decision affected. We therefore use public Finance Agent Benchmark questions as a realistic workload, not as our evaluation target, and report task accuracy only as context.

Retrieval-augmented, agentic and hybrid RAG. Retrieval-augmented generation grounds answers in retrieved documents [31]. Agentic RAG lets an agent plan retrieval iteratively, often by decomposing a question into sub-queries [32], [34]; hybrid RAG combines vector and knowledge-graph retrieval and has been applied to financial transcripts [33]; enterprise RAG frameworks add connectors, retrieval-time access control and citation display. These architectures decide *what to retrieve and how to answer*. FAOGF is orthogonal and architecture-agnostic: it sits at the tool and model boundary of any of them and records what was retrieved, what remained in context, under whose authority, and what action followed. RAG citations are the system's own claim about its sources; FAOGF checks that claim against an independent record—in our real-filing study it caught one answer citing a filing the agent never read.

Summary and gap. Table I compares FAOGF with each class of related work on the four properties stated in Section I plus runtime enforcement. To our knowledge, no prior system combines context-exact evidence, authority and policy binding, completeness under real OpenTelemetry loss, and content-free records—and none evaluates these properties on a real OTel pipeline and real financial filings.

TABLE I. Capabilities of related approaches (✓ = provided, ◐ = partial, — = not provided).

Approach	Context-exact evidence	Authority and policy version	Complete under span loss	No raw content retained	Runtime enforcement				
OTel GenAI tracing [1]	—	—	—	✓	—				
LLM observability with content logging [8], [9]	◐ prompts logged	—	—	—	—				
Delegation telemetry [5]	—	◐ authority only	—	✓	—				

Approach	Context-exact evidence	Authority and policy version	Complete under span loss	No raw content retained	Runtime enforcement				
Provenance models (W3C PROV) [10]	◐ expressible, not captured	◐ expressible	—	✓	—				
Tamper-evident logs [17], [18], [19], [20]	—	—	◐ for what is logged	✓	—				
RAG, agentic and hybrid RAG [31], [32], [33], [34]	◐ self-reported citations	◐ retrieval access control	—	—	—				
Financial QA benchmarks and LLMs [11], [24], [25], [26], [27], [28], [29], [30]	—	—	—	—	—				
Information-flow control for agents [22], [23]	◐ taint inside the agent	—	—	✓	✓				
FAOGF	✓	✓	✓	✓	✓				

IV. Problem Statement, Threat Model and Audit Queries

A. System Model

A *principal* (an analyst or service) submits a request to an orchestrator agent. Agents act only through tools—filing search and retrieval, XBRL facts, web search, calculation, delegation and report distribution—and through model calls. Each model call sees a bounded context window: tool results enter it and are later evicted. A *decision* is an agent's final answer; a *protected action* is one with external effect (here, distribute_report). Filings are versioned: an amended filing supersedes its original.

B. Threat Model

We consider four sources of audit failure. (1) **Lossy telemetry:** sampling, queue overflow, exporter outages and collector processors drop or rewrite spans; this is the default behavior of production pipelines, not an attack. (2) **Manipulated agents:** untrusted content (web results, third-party notes) may carry instructions, and the model may act on them. (3) **Unreliable self-report:** agent code may omit or misstate what it read, so audit facts must be bound at a boundary the agent cannot bypass. (4) **Record tampering:** an insider with write access may edit, delete, reorder or rewrite stored records. Out of scope are a compromised gateway host, compromised model weights, and collusion between the operator and the external anchor.

C. Audit Queries

We distilled nine questions an auditor or supervisor would ask after an incident. Table II defines each and its metric. Q1–Q7 are answerability queries scored against an oracle; Q8–Q9 measure whether the system knows its own record is incomplete.

Q3 is defined on *context*, not on retrieval: a filing the agent read but evicted before deciding did not support the decision. This choice is what separates evidence provenance from a log of tool calls, and it is where existing telemetry most often errs.

TABLE II. Audit queries.

ID	Audit question	Metric				
Q1	Under whose authority, through which delegation chain, did each action run?	Accuracy				
Q2	Which protected actions were attempted, with what outcome, under which policy version?	Recall; version accuracy				
Q3	Which filings, at which version, were in the model's context when each decision was made?	Precision / recall				
Q4	A filing was restated: which decisions relied on the superseded version (blast radius)?	Precision / recall				
Q5	Which actions executed after the acting delegation, or an ancestor, was revoked?	Recall				
Q6	Which actions were evaluated under a policy version older than the one in force?	Recall				
Q7	Which protected actions were requested by a model call whose context held untrusted content?	Precision / recall				
Q8	What share of lost spans can the auditor detect as lost?	Detection rate				
Q9	What share of records whose audit attributes were stripped in transit is detected?	Detection rate				

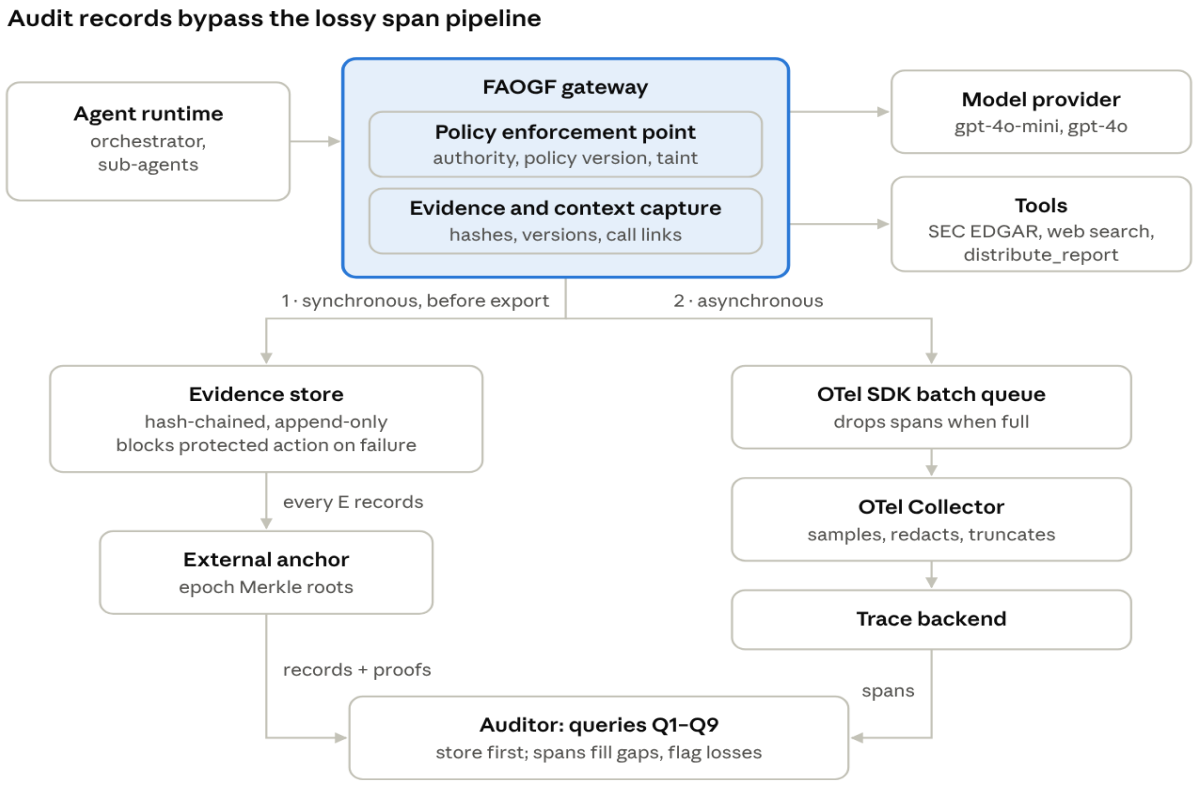


Figure 1. Component-level architecture. Every model and tool call passes through the gateway, whose policy enforcement point binds authority, policy version and taint before the call proceeds, and whose capture component records evidence and context. Each governed operation is written to the hash-chained evidence store before its span is exported. Sampling, queue overflow and collector processors act only on the span path; the auditor answers from the store and uses spans to detect what that path lost.

V. FAOGF Design

FAOGF is a capture boundary—a gateway—through which every model call and tool call of a governed agent passes. The gateway, not the agent, mints identifiers, binds authority, evaluates policy and writes records, so an agent cannot omit or misstate what it read. Fig. 1 shows the two paths every governed operation takes.

A. Evidence Records

When a tool returns content, the gateway writes an evidence record: a fresh evidence identifier, the source identifier (an EDGAR accession number or a synthetic filing id), the source version, the SHA-256 hash of the exact bytes returned, a trust class (authoritative_filing, derived or untrusted_web) and a timestamp. The span carries the identifiers and a compact descriptor source@version#hash-prefix|trust. No raw content is stored on either path; an auditor holding the source can verify the hash.

B. Context Binding

The central mechanism is the binding between a model call and its context. Before each model call, the runtime reports which evidence identifiers are currently in the model's context window; the gateway records them in a model-call record and flags any identifier it did not itself issue. Each tool call carries the identifier of the model call that requested it. Each decision record lists the evidence in context at the final model call plus the identifiers of sub-agent decisions it consumed. These three links—evidence → model call → action or decision—let an auditor answer Q3, Q4 and Q7 from what the model could actually attend to, rather than from what the agent had ever retrieved.

C. Authority and Policy Binding

FAOGF adopts delegation semantics from [5]: each delegation names a principal, a parent, a scope that may only narrow, and an expiry, and can be revoked with its descendants. Policies are declarative, versioned and evaluated with deny-overrides and default deny. Every action record carries the delegation identifier and lineage, the policy identifier, version and matching rule, the decision, and the human-approval outcome. Q1, Q2, Q5 and Q6 follow from these fields.

D. Dual-Path Capture and the Fail-Closed Protocol

Every governed operation is written to an append-only evidence store before its span is exported. Each record is chained to its predecessor:

$$h_i = \mathrm{SHA256}(seq_i \parallel kind_i \parallel ts_i \parallel body_i \parallel h_{i-1})$$

The span then carries a pointer to its record (fin.audit.record.id) and a record status. The protocol differs by risk:

1. Bind authority from the gateway-held delegation; reject unknown or revoked delegations.
2. Evaluate the policy version in force; deny if evaluation fails.
3. For a **protected** action, durably write an intent record. If the write fails, block the action (fail-closed).
4. Execute, then write the action record with outcome, policy and approval fields.
5. For a **read**, a failed store write does not block; the span is marked unrecorded so the gap is visible.
6. Optionally, write to a local write-ahead spool when the store is unreachable and replay it later, which keeps protected actions available during store outages.

Audit queries read the store first and fall back to spans only for fields the store lacks. Because every record names the span it should have produced, the auditor can detect lost spans (Q8) and spans that arrived without their audit attributes (Q9).

E. Taint-Aware Policy

Because the gateway knows each model call's context, it can evaluate the predicate *untrusted content was in the context of the model call that requested this action*. Policies may deny or escalate to a human on this predicate. A coarser variant available to delegation-only telemetry marks an action tainted if any earlier step in its delegation lineage retrieved untrusted content; we compare both.

F. Epoch Anchoring and Selective Disclosure

A hash chain detects single-point edits but not a consistent rewrite of a whole suffix. Every E records, FAOGF computes a Merkle root over the epoch's record hashes and publishes it to an external anchor such as a timestamping service, a transparency log or a regulator inbox. An auditor can then verify any one record with a logarithmic-size inclusion proof without receiving the rest of the store, and a rewrite is localized to its epoch.

G. Semantic-Convention Registry

The fin.audit.* vocabulary—19 attributes and three span groups (execute_tool, chat, decision) with required and conditionally required fields—is published as an OTel Weaver registry and validated with weaver registry check (v0.28.0). A conformance checker scores emitted spans against it, which turns pipeline misconfiguration into a measurable signal.

VI. Implementation

The prototype is about 3,400 lines of Python on the OpenTelemetry Python SDK 1.45. Table III lists its components.

The gateway emits standard GenAI spans and adds fin.audit.* attributes, so any OTel backend can store and query them. The store supports an in-memory backend for experiments and an fsync-on-append file backend; a production deployment would use a WORM-capable object store. Agents never call tools or the model directly; the runner reports the evidence identifiers in context before each model call, and the gateway flags any identifier it did not issue (none occurred in our runs).

TABLE III. Prototype components.

Component	Role				
faogf/gateway.py	Capture boundary: authority binding, policy evaluation, fail-closed protocol, evidence and context records, span emission				
faogf/store.py	Append-only hash-chained store (memory and file backends) and write-ahead spool				
faogf/delegation.py, faogf/policy.py	Narrowing delegations with lineage and revocation; versioned deny-overrides policies with escalation and taint predicates				
faogf/merkle.py	Epoch Merkle roots, inclusion proofs, anchored-root checking				
faogf/conformance.py, semconv/registry/	Weaver-format fin.audit.* registry and span conformance checker				
agent/runner.py	Orchestrator and analyst agents with a bounded context window, delegation and loop guard				
agent/edgar.py	Live SEC EDGAR tools: company lookup, filing list, document reader, XBRL company facts, with a disk cache				
agent/llm.py	Deterministic scripted model and OpenAI client with retries and a response cache				

Component	Role				
experiments/	Workload generator, audit queries Q1–Q9 under every condition, OTel Collector harness, E1–E15 driver, real-filing study				

VII. Experimental Setup

A. Workloads

Controlled study. A synthetic corpus of 20 fictional issuers with annual, quarterly and current reports, restatements that supersede earlier versions, and web pages and analyst notes carrying planted instructions in five styles. A task generator draws requests that read filings, search the web, delegate analysis to a sub-agent, compute and distribute reports; revocations, a mid-run policy change (v1 → v2) and a stale-policy gateway are injected. A deterministic scripted model drives the agents so that every condition sees identical behavior. Each configuration runs 200 workflows for each of five seeds (7–11); tables report mean ± 95% confidence interval across seeds.

Real-filing study. Agents run on gpt-4o-mini (all 50 public Finance Agent Benchmark questions [11], two runs) and gpt-4o (first 20 questions) against live SEC EDGAR, with temperature 0, a six-message context window per agent and a 25-step limit. In 64–70% of workflows the agent also receives a third-party note containing an injected instruction. Ground truth for Q3 and Q7 comes from the runner's own record of context contents, independent of any telemetry condition.

All nine audit queries were evaluated in the controlled study; the real-filing study scored Q3 and Q7. Table XVII summarizes which audit queries each study evaluated.

B. Telemetry Conditions

All conditions are *projections* of the same runs: the gateway emits the richest telemetry once, and each condition keeps only what its design would capture. Differences therefore come from what was captured, not from run-to-run variation (Table IV).

Baselines answer each query with the best heuristic their data permits: for example, B0 and B1 attribute to a decision every filing read in the workflow's reachable traces, and B3 parses the final model call's logged input, excluding evicted messages. Ablations remove one FAOGF component each: version identity, context binding, policy fields, delegation lineage, the model-call link, or the store.

TABLE IV. Telemetry conditions.

Condition	What is captured	
B0	OTel GenAI spans; tool arguments on, tool results and prompts off	
B1	B0 + delegation identifiers, principal and parent (approximates [5])	
B2	B0 + full tool results	
B3	B2 + full model input messages at every call	
FAOGF-S	All fin.audit.* span attributes; spans only, no store	
FAOGF	FAOGF-S + hash-chained evidence store (store first, span fallback)	

TABLE V. Controlled experiments.

ID	Question	Varied				
E1	Answerability without faults	Sub-agent hand-off: inline, linked, fragmented				
E2	Span-export loss	Drop rate 0–50%				
E3	Store write failures	Fail rate 0–20%				
E4	Tamper detection	Edit, delete, insert, reorder, suffix rewrite				
E5	Per-call overhead	Bare tool, GenAI span, FAOGF memory/file store				
E6	Ablations	One component removed				
E7	Scale	250–10,000 workflows				
E8	Correlated outages	Outage share 0–25%, with and without spool				
E9	Sensitivity	Context eviction 0–90%; delegation rate 0–80%				
E10	Content-logging baselines	B2, B3 vs FAOGF; linked, fragmented, 30% loss				
E11	Enforcement	Six policies; injections to allowed internal lists				
E12	Merkle anchoring	Epoch size 64–1,024				
E13	OTel sampling	Head 100–1%; Collector tail sampling				
E14	Export-queue overflow	SDK queue 2,048–32 over OTLP to a Collector				
E15	Collector processors	Pass-through, redaction allow-list, argument deletion, truncation				

C. Experiments

Table V lists the fifteen controlled experiments. E13 (tail sampling), E14 and E15 export spans with the SDK's BatchSpanProcessor over OTLP/HTTP to otelcol-contrib v0.163.2, whose file exporter output is parsed back for analysis. They were run on two machines: a cloud Linux container (x86-64) and an Apple M-series laptop. The exact Collector configurations are released with the artifact (Section XIII).

D. Metrics

Precision and recall are computed per query against the oracle; a prediction set that is empty when the truth is non-empty counts as a recall failure and leaves precision undefined. Storage is measured as serialized attribute and record bytes per workflow; retained sensitive content as raw content bytes and planted client account identifiers found in telemetry. Overhead is wall-clock time added per call, excluding tool work. Real-filing answers are graded by an LLM judge (gpt-4o-mini) against the benchmark's public rubric; we report accuracy only as context.

VIII. Results

In the controlled study, FAOGF answered every answerability query (Q1–Q7) exactly in every configuration we tested, and detected lost spans (Q8) and stripped audit attributes (Q9) at the rates reported below. In the real-filing study, where Q3 and Q7 were scored, it was exact on both (Table XVII). The baselines' accuracy depended on hand-off mode, context eviction, sampling rate, queue size, collector configuration and model behavior. We report results by research question.

TABLE VI. Audit-query answerability, fragmented hand-off, no faults (mean ± 95% CI over 5 seeds; n/a = not

expressible).

Query	B0	B1	FAOGF-S	FAOGF	
Q1 authority chain (acc.)	n/a	1.00	1.00	1.00	
Q2 protected attempts (R)	1.00	1.00	1.00	1.00	
Q2 policy version (acc.)	n/a	n/a	1.00	1.00	
Q3 supporting filings (P)	0.80 ± 0.04	0.82 ± 0.03	1.00	1.00	
Q3 supporting filings (R)	0.88 ± 0.01	1.00	1.00	1.00	
Q3 version-exact sources (P)	n/a	n/a	1.00	1.00	
Q4 restatement blast radius (P)	0.40 ± 0.06	0.42 ± 0.05	1.00	1.00	
Q4 restatement blast radius (R)	0.85 ± 0.10	1.00	1.00	1.00	
Q5 post-revocation actions (R)	n/a	n/a	n/a	1.00	
Q6 stale-policy actions (R)	n/a	n/a	1.00	1.00	
Q7 untrusted-context exposure (P)	0.86 ± 0.06	0.86 ± 0.06	1.00	1.00	
Attribute bytes per span	135	223	538	538	
Store bytes per workflow	—	—	—	11,400 ± 422	

A. RQ1 — Answerability Without Faults

Table VI gives answerability with asynchronous (fragmented) sub-agent hand-off and no faults. Plain GenAI tracing (B0) cannot express authority, policy version or revocation at all. Where it can answer, it over-attributes: about one in five filings it names as supporting a decision had been evicted from context before the decision (Q3 precision 0.80), and 60% of the decisions it flags as exposed to a restatement were not (Q4 precision 0.40). Adding delegation identifiers (B1) restores recall lost to fragmented traces but leaves precision unchanged, because delegation says who acted, not what they saw. Span-only FAOGF (FAOGF-S) matches the full system when no data is lost.

The baselines' errors grow with the behavior that creates them (Table VII). As the share of workflows that evict content rises from 0 to 90%, B0's Q3 precision falls from 1.00 to 0.39; as the share that delegate rises to 80%, its recall falls to 0.77. FAOGF stays at 1.00 throughout.

Ablations. Every FAOGF component is the sole support for at least one query (Table VIII). Removing version identity collapses blast-radius precision to 0.49; removing context binding returns Q3 precision to baseline (0.82); removing lineage drops post-revocation recall to 0.69; removing the model-call link returns exposure precision to 0.86; removing policy fields makes Q2-version and Q6 unanswerable.

TABLE VII. Sensitivity of baselines to agent behavior (Q3 filings).

Share of workflows	B0	B1	FAOGF	
Evicting context: 30% (P)	0.80	0.82	1.00	
Evicting context: 60% (P)	0.61	0.65	1.00	

Share of workflows	B0	B1	FAOGF	
Evicting context: 90% (P)	0.39	0.45	1.00	
Delegating: 40% (R)	0.88	1.00	1.00	
Delegating: 80% (R)	0.77	1.00	1.00	

TABLE VIII. Ablations (FAOGF minus one component).

Query	Full	−version	−context	−policy	−lineage	−model-call link	
Q3 version-exact (P)	1.00	n/a	0.83	1.00	1.00	1.00	
Q3 filings (P)	1.00	1.00	0.82	1.00	1.00	1.00	
Q4 blast radius (P)	1.00	0.49	0.84	1.00	1.00	1.00	
Q2 policy version	1.00	1.00	1.00	n/a	1.00	1.00	
Q5 post-revocation (R)	1.00	1.00	1.00	1.00	0.69	1.00	
Q6 stale policy (R)	1.00	1.00	1.00	n/a	1.00	1.00	
Q7 exposure (P)	1.00	1.00	0.96	1.00	1.00	0.86	

B. RQ2 — Real OpenTelemetry Pipelines

Sampling. Head sampling discards whole traces, so every span-based design loses audit coverage in proportion to the sampling rate (Table IX). At a 10% rate, B0, B1 and FAOGF-S recover 9–12% of protected attempts and 10–12% of authority chains; FAOGF recovers all of them. Tail sampling that keeps every trace containing an error or a distribute_report call preserves Q2 and Q7 recall, but still loses about half of supporting filings and authority chains, because the evidence behind an action often lives in a different trace from the action itself.

The consequence is practical: with FAOGF, an operator can sample traces at 1% for observability cost while audit completeness is unchanged, because the evidence store does not sample.

Export-queue overflow. We exported spans through the SDK's batch processor over OTLP to a real Collector under burst load (Table X). How much the queue dropped depended on the machine: with a 512-span queue, 0.9% of spans were lost on the Linux container and 31% on the faster M-series laptop, where spans were produced faster than the exporter drained them. Span-based audit recall tracked the loss on both machines; FAOGF stayed at 1.00 and detected 88–97% of the lost spans, against 0.2–8% for orphan-based heuristics. An operator cannot bound this loss in advance, which is the case for an audit path that does not depend on it.

Collector processors. Table XI shows that two common hardening steps silently break audit telemetry. A redaction processor configured as an allow-list of standard semantic-convention keys deletes every fin.audit.* attribute: span-only FAOGF falls to 0.12 Q3 recall and 0.00 authority accuracy, and nothing in the backend reveals the loss. Full FAOGF still answers every query from its store, detects the stripping on 100% of affected spans (Q9), and the registry conformance check drops from 1.00 to 0.00. Deleting tool arguments for privacy blinds B0 and B1 on supporting filings (Q3 recall 0.12) but does not affect FAOGF, which never relies on raw arguments. Truncating attribute values to 64 characters had no effect on any design.

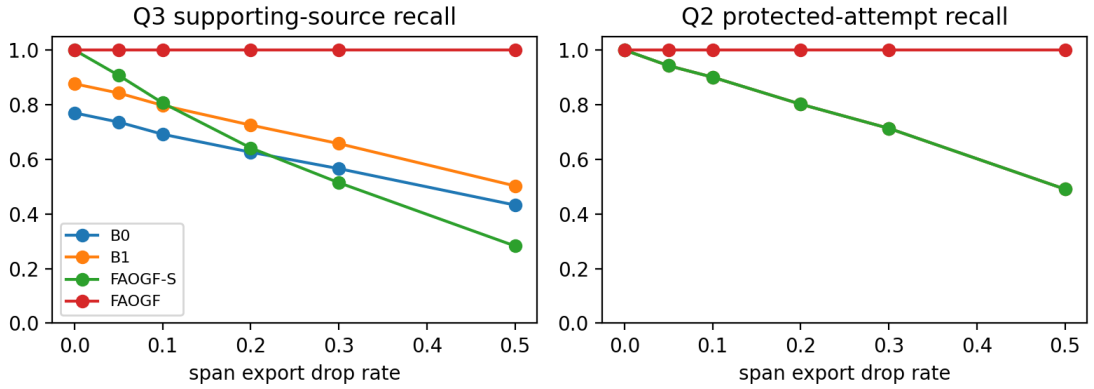


Figure 2. Supporting-source recall (left) and protected-attempt recall (right) as the span-export drop rate rises from 0 to 50%, controlled study, 5 seeds. B0, B1 and FAOGF-S coincide in the right panel. FAOGF stays at 1.00 because it answers from the evidence store; span-only FAOGF degrades fastest because its richer spans are its only copy.

Simulated span loss, store faults and outages. The in-process experiments agree. With 50% of spans dropped at random, B0, B1 and FAOGF-S keep 0.43, 0.50 and 0.28 of supporting sources respectively—FAOGF-S degrades fastest because its richer spans are its only copy—while FAOGF keeps 1.00 and detects 97% of lost spans versus 10% for orphan detection (Fig. 2). With up to 20% of store writes failing, no protected action executed without a durable record; 9.4 of 97.8 protected attempts per run were blocked and up to 441 read spans were flagged unrecorded. Under correlated outages that hit spans and store together (25% of the run), FAOGF without a spool fell to 0.79 Q3 recall; a local write-ahead spool restored 1.00 with 770 records spooled and no actions blocked.

TABLE IX. Audit recall under OTel sampling (Linux; Apple M-series in parentheses where it differs by more than 0.02).

Policy	Spans kept / workflow	Q1 auth. (B1)	Q2 attempts (B1)	Q3 filings R (B1)	FAOGF, all queries	
Head 100%	14.3	1.00	1.00	1.00	1.00	
Head 50%	7.3	0.52	0.50 (0.52)	0.53	1.00	
Head 10%	1.4–1.6	0.12 (0.10)	0.12 (0.09)	0.20 (0.18)	1.00	
Head 1%	0.1	0.01	0.01	0.11	1.00	
Tail: errors + sends + 10%	7.2	0.53	1.00	0.51	1.00	

TABLE X. SDK queue overflow over OTLP to otelcol-contrib (Linux / M-series).

Queue size	Spans dropped	Q2 attempts R, B1	Q3 filings R, B1	FAOGF, all queries	Lost spans detected, FAOGF	
2,048	0% / 0%	1.00 / 1.00	1.00 / 1.00	1.00	—	
512	0.9% / 31%	1.00 / 0.71	0.99 / 0.72	1.00	0.97 / 0.88	
128	4.8% / 78%	0.97 / 0.23	0.95 / 0.31	1.00	0.92 / 0.88	
32	8.0% / 94%	0.92 / 0.05	0.94 / 0.17	1.00	0.94 / 0.88	

TABLE XI. Collector processors (identical on both machines).

Processor	Registry conformance	Q1 auth., B1 / FAOGF-S	Q3 filings R, B1 / FAOGF-S	FAOGF, all queries	Stripping detected, FAOGF	
Pass-through	1.00	1.00 / 1.00	1.00 / 1.00	1.00	—	
Redaction allow-list	0.00	0.00 / 0.00	0.88 / 0.12	1.00	1.00	
Delete tool arguments	1.00	1.00 / 1.00	0.12 / 1.00	1.00	—	
Truncate to 64 chars	1.00	1.00 / 1.00	1.00 / 1.00	1.00	—	

C. RQ3 — Logging Everything Instead

The strongest alternative to an audit layer is to log every tool result (B2) or every tool result and full prompt (B3). Table XII shows that B3 is a serious competitor on evidence questions when traces are linked: it reconstructs the final model call's context from the logged prompt and matches FAOGF on Q3, Q4 and Q7. It fails in three ways. It inherits the span path's fragility: with asynchronous hand-off its Q3 recall falls to 0.85, and with 30% span loss to 0.78, with 71% of protected attempts found. It cannot answer authority, policy-version or revocation questions, because prompts do not contain them. And it retains raw content: on average 6.7 planted client account identifiers per workflow, against none for FAOGF.

Storage depends on document size. With the short synthetic filings, B3 used 9.4 KB per workflow against FAOGF's 15.1 KB. On real filings the order reverses sharply: the prompts gpt-4o-mini and gpt-4o actually received—exactly what B3 must store—amounted to 344 KB and 883 KB per workflow, against 47 KB and 85 KB for FAOGF's spans and store combined (7.3× and 10.4× less). Because every model call re-sends the growing conversation, full-prompt logging grows faster than linearly with agent steps, while FAOGF grows linearly.

TABLE XII. Content-logging baselines (controlled study).

Metric	B2	B3	FAOGF	
Q3 filings P / R, linked	0.82 / 1.00	1.00 / 1.00	1.00 / 1.00	
Q3 filings R, fragmented	0.88	0.85	1.00	
Q3 filings R, 30% span loss	0.65	0.78	1.00	
Q2 attempts R, 30% span loss	0.71	0.71	1.00	
Q4 blast radius P, linked	0.83	1.00	1.00	
Q1, Q2 version, Q5, Q6	n/a	n/a	1.00	
Raw content retained, KB / workflow	0.7	6.5	0	
Client account IDs retained / workflow	2.0	6.7	0	

D. RQ4 — From Audit to Enforcement

In E11, planted instructions told the agent to send reports to internal distribution lists (all-staff@, board-pack@) that the domain allow-list permits; a susceptible scripted agent obeyed them. Table XIII compares six policies over five seeds. The allow-list alone executed all 32.8 harmful sends per run. Requiring human approval for every send blocked them, at a cost of 94.6 approvals per run. Escalating only sends whose requesting model call had untrusted content in context—a predicate available

only through context binding—also blocked every harmful send and blocked no legitimate one, with 59.2 approvals: 37% fewer. Denying instead of escalating needed no approvals but blocked 26 legitimate sends per run. This result comes from the controlled study's simulated susceptible agent, which is scripted to obey planted instructions, and compares context-taint escalation with the allow-list plus approve-every-send policy (Table XIII). It does not measure enforcement on live models: in the real-filing study neither model followed any injected instruction (Section VIII-G), so no harmful send occurred for a policy to stop.

Context taint improves on the coarser lineage taint, but modestly: 4% fewer approvals at default settings, rising to 8% fewer approvals and 17% fewer wrongful blocks when 90% of workflows evict content. The large gain comes from escalating selectively at all; precise context binding refines it.

TABLE XIII. Enforcement against injections that target allowed internal destinations (simulated susceptible agent; per run of 200 workflows).

Policy	Harmful sends executed	Legitimate sends blocked	Human approvals	
Allow-list only	32.8	0	0	
Allow-list + approve every send	0	0	94.6	
Taint-deny, lineage taint	0	28.6	0	
Taint-deny, context taint (FAOGF)	0	26.4	0	
Taint-escalate, lineage taint	0	0	61.4	
Taint-escalate, context taint (FAOGF)	0	0	59.2	

E. Integrity and Selective Disclosure

Across 500 single-point manipulations of a 3,555-record chain—125 deletions, 120 body edits, 127 forged insertions and 128 reorderings—chain verification detected all 500. A consistent rewrite of the whole chain suffix passed chain verification, as expected, and was detected by every epoch-anchoring configuration and localized to the correct epoch (Table XIV). An auditor can verify a single record with about 1 KB of proof and record in 7–12 µs, disclosing about 0.05% of the store.

F. Overhead and Scale

FAOGF adds about 0.2 ms per call at the median: 194 µs with a durable file store against 38 µs for a plain GenAI span (Table XV), negligible beside model calls of hundreds of milliseconds. At 10,000 workflows (144,368 spans, 115 MB store), verifying the full chain took 2.1 s and a complete audit sweep of all queries took 521 ms, against 392 ms for B0, which answers fewer of them.

TABLE XIV. Epoch Merkle anchoring (3,555 records, 2.4 MB store).

Epoch size	Anchors (bytes)	Proof + record (bytes)	Verify (µs)	Suffix rewrite: chain / anchors	Localized	
64	56 (3,584)	1,028	7.1	missed / detected	yes	
256	14 (896)	1,178	9.7	missed / detected	yes	
1,024	4 (256)	1,327	12.3	missed / detected	yes	

TABLE XV. Per-call overhead, tool work excluded (µs).

Path	p50	p95	p99	
Bare tool call	1.1	1.5	2.0	
OTel GenAI span	37.7	70.4	91.0	
FAOGF read, memory store	141.5	203.6	319.7	
FAOGF read, file store	196.7	287.5	562.1	
FAOGF protected action, file store	193.4	291.3	661.2	

G. Real-Filing Study

On live SEC filings we scored the two audit queries the live workload exercises: supporting filings (Q3) and untrusted-context exposure (Q7) (Table XVI). Q1, Q2 and Q4–Q6 were not scored because the live workload injected no revocations, restatements or policy changes, and Q8–Q9 do not apply because no pipeline loss was injected. On Q3 and Q7 the controlled findings held. Standard tracing named supporting filings that were no longer in the model's context in 17–18% of cases with gpt-4o-mini and 4.5% with gpt-4o; FAOGF was exact in all runs. The gap between models is itself informative: gpt-4o's answers rested on recently read filings, so less had been evicted, exactly as Table VII predicts. An auditor using standard tracing cannot know in advance which regime a given model is in. Exposure attribution (Q7) was weaker still for the baselines, at 0.25–0.38 precision, because a planted note entered the context early and was often evicted before the agent acted. Every run's evidence store verified intact, and we re-verified the two exported stores independently.

Two further observations matter. First, neither model followed any of the 78 injected instructions (32 for gpt-4o-mini with a warning, 32 without, 14 for gpt-4o), across five styles and with or without a warning in the system prompt. The planted note was nevertheless in context for 28–36% of model calls. The live study therefore measures exposure, not enforcement: exposure still had to be proven, and only FAOGF proved it precisely (Q7 precision 1.00 against 0.25–0.38). Second, FAOGF caught one answer citing a filing the agent never read—a fabricated citation that the answer text alone does not reveal. Task accuracy is reported as context only: our claims concern auditability, not agent skill, and with 20–50 questions the accuracy estimates carry wide intervals.

TABLE XVI. Real-filing study on public Finance Agent Benchmark questions with live SEC EDGAR.

Metric	gpt-4o-mini, warning	gpt-4o-mini, no warning	gpt-4o, no warning	
Questions	50	50	20	
Q3 filings P: B0 / B1 / FAOGF	0.82 / 0.82 / 1.00	0.83 / 0.80 / 1.00	0.96 / 0.96 / 1.00	
Q3 filings R: B0 / B1 / FAOGF	0.96 / 1.00 / 1.00	0.95 / 1.00 / 1.00	1.00 / 1.00 / 1.00	
Q7 exposure P: B0 / FAOGF	0.30 / 1.00	0.38 / 1.00	0.25 / 1.00	
Injections obeyed	0 / 32	0 / 32	0 / 14	
Model calls with planted note in context	—	219 / 612	92 / 334	
Citations to filings in context / evicted / never read	98.7% / 1.3% / 0%	98.8% / 0% / 1.2%	100% / 0% / 0%	
Telemetry KB / workflow: full prompts vs FAOGF	—	344 vs 47	883 vs 85	
Answered within step limit	39 / 50	43 / 50	15 / 20	

Metric	gpt-4o-mini, warning	gpt-4o-mini, no warning	gpt-4o, no warning	
Task accuracy (LLM judge)	20%	16%	40%	

IX. Discussion

Audit and observability have opposite loss tolerances. Observability pipelines are built to keep a representative subset; audit needs every protected action. Every real mechanism we tested—head and tail sampling, queue overflow, attribute processors—is defensible for observability and silently destructive for audit. FAOGF does not ask operators to give these up. It moves the audit guarantee to a path that does not pass through them, and uses the span path to detect when the observability copy is incomplete.

What the model saw is the right unit of evidence. Logging what an agent retrieved overstates what supported a decision whenever context is bounded, and the overstatement grows with eviction, delegation and model behavior that an auditor cannot observe. Binding evidence to the model call is a small mechanism—one list of identifiers per call—that removes this whole class of error, and the ablations show no other component substitutes for it.

Hashes, not content. The design keeps the auditor able to prove which bytes were in context without the firm retaining raw filings or client data in telemetry. On real filings this was also the cheaper option by 7–10×, because conversation-level logging re-stores the same content at every model call.

One record, two uses. The binding built for after-the-fact audit is also a runtime signal. On a simulated susceptible agent, escalating only actions taken with untrusted content in context reduced approval load by 37% against approving every send, without letting an attack through (Table XIII). The live models did not act on injected instructions, so this enforcement result has not yet been observed on a live model. Prompt-level defenses and information-flow designs [22], [23] are complementary: they reduce how often an agent is manipulated, and FAOGF records and gates the cases that remain.

Why neither model obeyed. Both OpenAI models resisted every injection style, with and without a warning. We do not read this as evidence that injection is solved: the attacks were planted in a clearly labeled notes tool, and stronger or better-hidden attacks exist [21]. It does show that exposure is common even when compliance is not—the planted note was in context for 28–36% of model calls—and that proving exposure is a separate requirement from preventing compliance.

X. Limitations and Threats to Validity

Internal validity. The controlled study uses a scripted model, so agent behavior is a parameter rather than an observation; we mitigate this with sensitivity sweeps (Table VII) and the real-model study, whose findings agree. Ground truth comes from the runner, which is part of our harness; a bug there would bias every condition equally, but could still mis-score all of them. We found and fixed one such bug, in which the scripted agent re-sent a legitimate report, and re-ran every experiment.

Construct validity. Context binding records what was available to the model, not what it relied on. A filing in context may have had no influence on an answer; counterfactual attribution—re-running a call with one item removed—would measure influence and is future work. Context taint is precise for direct exposure but does not propagate through text the model wrote while exposed; lineage taint covers that case more coarsely.

External validity. The real-filing study covers 50 public questions, two models from one provider, and one agent design, and scores two of the nine audit queries (Q3, Q7). Neither model obeyed any injection, so the enforcement result (Table XIII) rests on a simulated susceptible agent. Our Collector experiments use one Collector version and four processor configurations; production pipelines vary. Queue-overflow rates are hardware-dependent, as the two machines show, which is why we report both. The models were called by alias, and the harness did not record a dated snapshot identifier (Table XVIII).

Security assumptions. FAOGF trusts the gateway host and the runtime's report of context contents; a compromised runtime could under-report context. Epoch anchoring detects store rewrites only if the anchor is outside the operator's control. We did not evaluate denial of service against the store, nor retention and deletion obligations such as data-subject erasure.

XI. Implications for Practice

- **Do not treat traces as the audit record.** If a supervisory or record-keeping process depends on spans, it inherits every sampling and pipeline decision made for cost. Keep a separate, synchronous record for governed operations and use traces for observability.

- **Record context, not just tool calls.** For any agent with a bounded context window, "what it read" overstates "what supported the answer." A list of evidence identifiers per model call is enough.

- **Fail closed on protected actions, open and flagged on reads.** This kept every protected action recorded under 20% store failure while losing no read availability; a local spool removes most of the availability cost.

- **Test collector configurations against an audit registry.** A one-line redaction allow-list deleted all audit attributes in our runs. A conformance check against a published registry turns that into an alert.

- **Escalate on exposure instead of approving everything.** In our simulated susceptible-agent experiment, gating sensitive actions on untrusted content in context cut human approvals by 37% at equal protection, compared with approving every send.

- **Give auditors proofs, not databases.** Anchored epoch roots let a firm prove one record to a regulator with about 1 KB of data.

XII. Conclusion

Auditing autonomous financial agents requires knowing which evidence versions were in a model's context, under which authority and policy an action ran, and whether untrusted content was present—and knowing it completely, even though the tracing pipelines that carry agent telemetry are designed to drop data. FAOGF binds decisions and actions to context-level evidence and records that binding on a synchronous, hash-chained path alongside standard OpenTelemetry spans. In fifteen controlled experiments, through a real OpenTelemetry Collector on two machines, FAOGF answered every answerability query (Q1–Q7) exactly and detected 88–98% of lost spans and every stripped audit attribute (Q8, Q9). Standard tracing degraded with sampling, queue overflow, collector configuration and model behavior, and full prompt logging cost 7–10× more storage and retained raw content. On live SEC filings with two models, where supporting filings (Q3) and untrusted-context exposure (Q7) were scored, FAOGF was exact on both while standard tracing misattributed 4–18% of supporting filings. In a simulated susceptible-agent experiment, the same binding drove a policy that stopped injected sends to allowed destinations with 37% fewer human approvals than approving every send; neither live model followed the injected instructions. Future work includes counterfactual evidence attribution, taint propagation through model-generated text, deployment behind the Model Context Protocol, and evaluation with more model families and stronger injection attacks.

XIII. Artifact Availability and Reproducibility

The FAOGF prototype, the fin.audit.* semantic-convention registry, the experiment drivers and the per-table result files are available at https://github.com/sowmithoriginal/faogf/tree/v1.0.0-ijdsml and archived with DOI 10.5281/zenodo.24181994. The release contains: (i) the prototype (Table III); (ii) the Weaver registry under semconv/registry/, validated with weaver registry check v0.28.0; (iii) the OpenTelemetry Collector configurations for every pipeline experiment under collector/ (otelcol-contrib v0.163.2): pass-through, tail sampling, redaction allow-list, tool-argument deletion and 64-character truncation; (iv) the SDK sampling and queue settings and the command that regenerates each table (repro/RUNS.md); (v) the scoring procedure, including oracle construction, precision and recall definitions, the empty-prediction rule, the confidence-interval method and the LLM-judge configuration (repro/SCORING.md); (vi) the recorded software environment (repro/environment-mac.json); and (vii) the complete model-call logs (requests and responses) of all three real-filing runs, and the evidence stores and exported spans of the two no-warning runs, so Table XVI can be re-scored and the stores re-verified without API access.

The controlled study is deterministic given seeds 7–11; tables report the mean over the five per-seed values ± the 95% Student-t interval. The real-filing study used, unmodified, the 50-question public validation split of the Finance Agent Benchmark [11], which the release redistributes under CC BY 4.0 as data/fab_public.csv (SHA-256 901e8cff… bbe05bff5). Table XVII lists which audit queries each study evaluated, and Table XVIII lists the models and run dates.

TABLE XVII. Audit-query coverage by study.

Query	Controlled experiments	Real filings	FAOGF			
Q1	E1, E2, E6, E10, E13, E15	—	exact			
Q2	E1–E3, E6, E10, E13, E14	—	exact			
Q3	E1, E2, E6, E9, E10, E13–E15	Table XVI	exact			
Q4	E1, E6, E10	—	exact			
Q5	E1, E6, E10	—	exact			
Q6	E1, E6, E10	—	exact			
Q7	E1, E6, E11	Table XVI	exact			
Q8	E2, E14	n/a	88–98% detected			
Q9	E15	n/a	100% detected			

TABLE XVIII. Real-filing study: models and run dates.

Run	Model (API alias)	Temperature	Questions	Model calls	Run date	
1, notes flagged as unverified	gpt-4o-mini	0	50	948	Oct. 24, 2026	
2, no warning	gpt-4o-mini	0	50	612	Oct. 24, 2026	
3, no warning	gpt-4o	0	20	334	Oct. 24, 2026	

Models were called through the OpenAI Chat Completions API by alias; the harness recorded the alias but not a dated snapshot identifier, so results reflect the snapshot each alias resolved to on the run date. Answers were graded by gpt-4o-mini against each question's public rubric. Software: Python 3.12.7, OpenTelemetry Python SDK 1.45.1, OpenAI Python SDK 3.18.0, otelcol-contrib v0.163.2, Weaver v0.28.0. Machines: x86-64 Linux cloud container; Apple silicon laptop.

Acknowledgment

The Finance Agent Benchmark public validation split is used, unmodified, under CC BY 4.0 from Vals AI [11].

References

1. OpenTelemetry, "Semantic conventions for generative AI systems," OpenTelemetry Specification, 2025. [Online]. Available: https://opentelemetry.io/docs/specs/semconv/gen-ai/

2. B. H. Sigelman *et al.*, "Dapper, a large-scale distributed systems tracing infrastructure," Google Technical Report dapper-2010-1, 2010.

3. OpenTelemetry, "OpenTelemetry Specification: Tracing API and SDK," 2025. [Online]. Available: https://opentelemetry.io/docs/specs/otel/

4. P. Las-Casas, G. Papakerashvili, V. Anand, and J. Mace, "Sifter: Scalable sampling for distributed traces, without feature engineering," in *Proc. ACM Symp. Cloud Computing (SoCC)*, 2019.

5. Mishra and Sharad, arXiv preprint arXiv:2606.09692, 2026.

6. W3C, "Trace Context," W3C Recommendation, 2021.

7. OpenTelemetry, "Weaver: OpenTelemetry semantic convention registry tooling," v0.28.0. [Online]. Available: https://github.com/open-telemetry/weaver

8. Langfuse, LangSmith, Arize Phoenix and OpenLLMetry product documentation, accessed Oct. 2026.

9. L. Dong, Q. Lu, and L. Zhu, "AgentOps: Enabling observability of LLM agents," arXiv:2411.05285, 2024.

10. L. Moreau and P. Missier, Eds., "PROV-DM: The PROV data model," W3C Recommendation, 2013.

11. A. Bigeard, L. Nashold, R. Krishnan, and S. Wu, "Finance Agent Benchmark: Benchmarking LLMs on real-world financial research tasks," arXiv:2508.00828, 2025. Public validation split (50 questions, CC BY 4.0), Vals AI. [Online]. Available: https://github.com/vals-ai/finance-agent/blob/main/data/public.csv. The exact file used is redistributed in the FAOGF artifact as data/fab_public.csv (SHA-256 901e8cff562819007d43ff0c10b43b2e386481eb40f2157ba18da8fbbe05bff5).

12. U.S. Securities and Exchange Commission, Rule 17a-4, "Records to be preserved by certain exchange members, brokers and dealers," 17 CFR 240.17a-4.

13. FINRA, Rule 3110, "Supervision."

14. Board of Governors of the Federal Reserve System and OCC, "Supervisory guidance on model risk management," SR 11-7, 2011.

15. FINRA, Regulatory Notice 24-09, "FINRA reminds members of regulatory obligations when using generative artificial intelligence and large language models," 2024.

16. Regulation (EU) 2024/1689 (Artificial Intelligence Act), Art. 12, "Record-keeping," *Official Journal of the European Union*, 2024.

17. B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Trans. Information and System Security*, vol. 2, no. 2, 1999.

18. [18] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging," in *Proc. USENIX Security Symp.*, 2009.

19. R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, 1988.

20. B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, IETF, 2013.

21. K. Greshake *et al.*, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proc. ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

22. E. Debenedetti *et al.*, "Defeating prompt injections by design," arXiv:2503.18813, 2025.

23. M. Costa *et al.*, "Securing AI agents with information-flow control," arXiv:2505.23643, 2025.

24. Z. Chen *et al.*, "FinQA: A dataset of numerical reasoning over financial data," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

25. F. Zhu *et al.*, "TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance," in *Proc. Annu. Meeting Assoc. Computational Linguistics (ACL)*, 2021.

26. Z. Chen *et al.*, "ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering," in *Proc. EMNLP*, 2022.

27. P. Islam *et al.*, "FinanceBench: A new benchmark for financial question answering," arXiv:2311.11944, 2023.

28. V. Reddy *et al.*, "DocFinQA: A long-context financial reasoning dataset," in *Proc. ACL*, 2024.

29. S. Wu *et al.*, "BloombergGPT: A large language model for finance," arXiv:2303.17564, 2023.

30. H. Yang, X.-Y. Liu, and C. D. Wang, "FinGPT: Open-source financial large language models," arXiv:2306.06031, 2023.

31. P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

32. A. Singh *et al.*, "Agentic retrieval-augmented generation: A survey on agentic RAG," arXiv:2501.09136, 2025.

33. B. Sarmah *et al.*, "HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction," arXiv:2408.04948, 2024.

34. D. Zhou *et al.*, "Least-to-most prompting enables complex reasoning in large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023.

35. V. Gupta *et al.*, "TempTabQA: Temporal question answering for semi-structured tables," in *Proc. EMNLP*, 2023.