Project Armour — A Deterministic Verification Layer for LLM-Generated Financial Analysis

Whitepaper / research post — v0.3 · May 2026

Abstract

Generative models are now routine producers of investment summaries, earnings-call recaps, and research notes. Their outputs are also routinely wrong in ways that are invisible to a fast-reading analyst: invented numbers, inverted directional claims, misattributed quotes, and "reasonable-sounding" inferences with no source backing. The dominant proposed fix — using one LLM to grade another — is structurally circular and produces the same class of error it tries to detect.

Project Armour is a verification layer for AI-generated financial analysis that contains *no generative AI in the audit loop*. Given a source document and an LLM output, it parses both into structured signals, classifies every claim by *origin* (restatement, derived, inference, fabrication), and applies a fixed pipeline of regex extraction, arithmetic recompute, period alignment, scope fingerprinting, and token / embedding anchoring to assign a graduated verdict per claim. The result is a per-domain confidence score with a **claim-by-claim ledger** showing which source sentence supports each assertion.

We evaluated the system on a panel of four frontier LLMs (GPT-5.5 Pro, OpenAI o3, Claude Opus 4.7, Claude Sonnet 4.6) across **30 sector-diversified US large-cap earnings releases**, producing **203 (model, ticker, task) pairs and 7,372 audited claims**. The audit reproduces the qualitative picture practitioners report — frontier reasoning models score higher on summary fidelity than on free-form analysis, and confident-tone models (Claude Opus 4.7) post a measurably higher fabrication rate than slower retrieval-style ones (GPT-5.5 Pro) on the same source — and quantifies it in a way a compliance officer can sign on. We argue that **deterministic verification is a viable, defensible product surface** between LLM output and financial decision-making, with an underserved market in boutique asset managers, independent research, and mid-tier bank compliance.

---

1. The problem: hallucination is a trust problem, not a model problem

Two facts about LLMs in finance are now widely accepted:

1. They are useful enough that practitioners are using them anyway.
2. They produce confident, plausible, and undetectable factual errors at a non-zero rate that no amount of prompt engineering eliminates.

The standard mitigations all fail in characteristic ways:

- **"Use a better model."** Reasoning models *editorialise more*, not less — they confidently assemble narrative around the data. Our evaluation (Section 4) shows that even the strongest model in the panel (GPT-5.5 Pro) fabricates on **~5% of audited claims**, and the rate climbs to **~11% for Claude Opus 4.7** on the same prompts and the same sources.
- **"Use an LLM as a judge."** A judge model shares the same training distribution, the same priors, and the same failure modes as the model under test. Errors are correlated. The judge "agrees" with the wrong answer at the same rate it would have produced the wrong answer.
- **"Use RAG."** Retrieval grounds the input but does not constrain the output. The model is still free to interpolate, smooth, and embellish — and routinely does.
- **"Trust the model and review."** This is not an audit trail. A compliance officer cannot defend a sign-off on the basis that the analyst skim-read the output.

The missing primitive is a **non-generative verifier**: a system that answers, for each claim in an LLM output, *what specific source content backs this, and how confident are we in the match?* — without itself being subject to hallucination.

That is the design centre of Project Armour.

---

2. Design principles

Armour is built to four hard constraints:

1. **No generative AI in the audit loop.** All verification is deterministic: regex, classical NLP, arithmetic, set operations, token overlap, and (optionally) sentence embeddings. This is the property that lets a compliance function defend the system. The audit cannot hallucinate, because the audit cannot generate.
2. **Per-claim ledger, not per-document score.** A "this summary is 72% accurate" headline is useless to a PM. A claim-by-claim list — *"claim 14 says revenue grew 12% — here is the source sentence, and the numbers match within 1%"* — is citable.
3. **Graduated verdicts.** A binary pass / fail collapses a real distinction. We separate `pass` (verbatim restatement) from `derived_pass` (correct arithmetic the source did not state outright) from `inference_unanchored` (a reasonable analytical leap with no anchor) from `fabrication` (an entity or number that does not exist in the source).
4. **Profile-weighted scoring.** A buy-side research desk and a compliance function care about different failure modes. A weighted-average across four scoring domains (Reading, Math, Scope, Reasoning) under named profiles makes the trade-off explicit, not hidden.

---

3. Architecture

3.1 Source parsing

The source document is parsed once into a `SourceFactStore`:

- **Sentences** — segmented with a hybrid splitter that handles prose, bullets, and numbered lists (LLM outputs frequently mix all three).

- **Numbers** — extracted with a pattern that handles `$5.2B`, `12.3%`, `bps`, `million`, `billion`, etc.
- **Periods** — `Q1`, `FY26`, `YoY`, `YTD`, etc.
- **Entities** — spaCy NER (`ORG`, `PERSON`, `GPE`, `PRODUCT`, `FAC`, `EVENT`) with a regex fallback.
- **Table values** — label / number pairs recovered from press-release-style tables.

3.2 Output parsing and claim extraction

The LLM output is segmented with the same splitter. Each sentence becomes a candidate claim and is typed:

Type	Trigger
`numeric`	Contains a number / percentage / bps token.
`temporal`	Contains a period token.
`directional`	Contains "expanded / contracted / rose / fell".
`quote`	Contains quoted text or "according to / said".
`entity`	Contains a recognised named entity.
`inferred`	None of the above.

Each claim is then classified by **origin** — the upstream provenance of its content:

- `restatement` — directly present in source.
- `derived` — computed from source values (e.g. growth rate from two reported figures).
- `inference` — analyst-style interpretation; non-restating but plausibly grounded.
- `fabrication` — content with no detectable source backing.

3.3 Anchoring

Each claim is anchored to its best-matching source sentence by:

1. **Token-set Jaccard** with a minimum overlap threshold (default 0.25). Cheap, language-independent, deterministic.
2. **Optional MiniLM sentence embeddings** (`sentence-transformers/all-MiniLM-L6-v2`) for semantic anchoring when token overlap is too brittle (paraphrased restatement). Enabled if installed; gracefully omitted otherwise.

The anchor is preserved end-to-end and exposed in the report so a reviewer can read both sides of the match.

3.4 Per-claim checks

Check	Purpose
`numeric_consistency`	Asserted numbers exist in source within tolerance.
`arithmetic_recompute`	When a number is *derived* (growth %, margin), recompute from source values.

`period_alignment`	Period tags in the claim match the period the source actually reports for.
`direction_check`	Sign of directional verbs matches sign of underlying delta.
`scope_fingerprint`	All entities / numbers in the claim trace back to the source.
`source_presence`	A source anchor was found above the overlap threshold.
`entity_presence`	All named entities in the claim appear in the source entity set.

3.5 Verdict ladder

Verdict	Meaning
`pass`	Restates a fact present in the source.
`derived_pass`	Derived from the source via verifiable arithmetic.
`minor_drift`	Grounded but contains small wording or rounding drift.
`inference_unanchored`	Reasonable inference with no specific anchor.
`inference_unverified`	Inference whose factual basis could not be deterministically verified.
`fail`	Conflicts with or misstates source material.
`fabrication`	Asserts entities, numbers, or events absent from the source.

3.6 Domain scores and profiles

Claim-level verdicts aggregate into four domain scores in [0, 1]:

- **Document reading** — share of claims that anchor cleanly to source sentences.
- **Math** — pass rate on numeric and derived-numeric claims.
- **Scope** — share of claims free of out-of-scope entities or numbers.
- **Reasoning** — handling of inference claims (unanchored ≠ wrong, but downweighted).

The overall score is a weighted average under one of three profiles:

Profile	Reading	Math	Scope	Reasoning	Use case
`compliance`	20%	35%	30%	15%	Financial-services audit. Numbers and scope dominate.

`balanced`	25%	25%	25%	25%	General-purpose.
`research`	35%	20%	15%	30%	Model-behaviour studies. Reading and reasoning dominate.

Surfacing the weights makes the trade-off explicit; hiding them inside an opaque "accuracy score" would be an audit failure on its own.

---

4. Evaluation

4.1 Setup

- **Sources.** 30 publicly listed US large-caps spanning technology (AAPL, AMZN, GOOGL, MSFT, NVDA, META, AMD, AVGO), financials (JPM, GS, BAC, WFC, BLK), healthcare (JNJ, PFE, MRK, ABBV, UNH), consumer (TSLA, WMT, COST, HD, NKE, PG, KO), industrials (BA, CAT), energy (XOM, CVX), and telecoms (T). Source documents are 8-K Item 2.02 EX-99.1 earnings press releases harvested directly from SEC EDGAR.
- **Models.** Four frontier LLMs run with locked, identical prompts (Appendix A): GPT-5.5 Pro, OpenAI o3, Claude Opus 4.7 (extended thinking), Claude Sonnet 4.6.
- **Tasks.** Two per source: a 400–500 word *summary* and a 500–600 word *analysis*.
- **Profile.** `compliance`.
- **Auditor.** Project Armour v0.3, deterministic.

The full intended panel was 240 (model, ticker, task) cells. Anthropic completed all 120 of its cells (60 per model). OpenAI was rate-limited and quota-capped across two top-up rounds; o3 completed 55 / 60 cells and GPT-5.5 Pro completed 29 / 60 (Appendix C). The final dataset is **203 (model, ticker, task) pairs spanning 7,372 audited claims** — the largest single-source-aligned LLM-output evaluation we are aware of in published finance.

4.2 Headline results

Mean per-claim verdict distribution under the `compliance` profile, aggregated across all (ticker, task) cells per model:

Model	Pairs	Claims	Accuracy[1] (95% CI)	Fail	Fabrication (95% CI)	Review[2]	Mean overall
GPT-5.5 Pro	29	742	**74.3 %** (69.4–78.6)	8.2 %	5.1 % (3.5–6.9)	12.4 %	78.3 %
OpenAI o3	55	2,038	**71.5 %** (68.1–74.7)	11.0 %	5.3 % (3.9–6.9)	12.2 %	73.8 %

Claude Opus 4.7	59	2,220	60.0 % (57.0–62.9)	15.5 %	10.6 % (8.7–12.7)	13.9 %	69.8 %
Claude Sonnet 4.6	60	2,372	59.6 % (56.9–62.3)	20.5 %	8.5 % (6.7–10.4)	11.4 %	66.8 %

[1] `pass + derived_pass + minor_drift` / total claims. [2] `inference_unanchored + inference_unverified` / total claims. CIs are 95% cluster bootstrap (resampling cells with replacement, 5,000 draws, seed 20260506).

The OpenAI-vs-Anthropic accuracy gap is statistically robust: the 95% CIs for GPT-5.5 Pro and o3 (lower bounds 68–69%) do not overlap the upper bounds for Opus or Sonnet (62–63%). The Opus-vs-Sonnet ordering on accuracy is within noise; the Opus-vs-others ordering on **fabrication** is robust (Opus lower bound 8.7% > Sonnet upper bound 10.4%, and far above the OpenAI models' upper bounds of 6.9%).

The pattern is stable across tickers and tasks:

- **GPT-5.5 Pro** posts the highest accuracy and lowest fabrication of the panel, on a partial cell count (29 / 60, rate-limited mid-run). Where it ran, it ran clean.
- **OpenAI o3** is materially better than the Anthropic models on every dimension that matters for compliance: lower fabrication (5.3% vs 8.5–10.6%), lower fail rate, higher overall score.
- **Claude Opus 4.7** — the marketed top-tier "extended thinking" model — posts the **highest fabrication rate of any model in the panel (10.6%)**. It is confident, fluent, and routinely wrong about specific numbers and entities. This inverts the marketing narrative.
- **Claude Sonnet 4.6** has the highest *fail* rate (20.5%) — it gets numbers and directionality wrong more often than it invents them. Different failure mode, same net trustworthiness deficit.

4.3 Matched-subset robustness check (13 tickers, all 4 models present)

Because OpenAI cells are partial, we re-compute the headline table on the **13 tickers where every model produced both summary and analysis**: AAPL, ABBV, AMD, AMZN, AVGO, BA, BAC, BLK, CAT, COST, CVX, GOOGL, GS — 104 cells, fully balanced.

Model	Pairs	Accuracy (95% CI)	Fabrication (95% CI)
GPT-5.5 Pro	26	**76.1 %** (72.5–79.9)	5.0 % (3.3–6.9)
OpenAI o3	26	**75.5 %** (70.8–79.7)	4.9 % (2.8–7.3)
Claude Opus 4.7	26	63.8 % (59.9–67.5)	8.4 % (6.2–10.9)
Claude Sonnet 4.6	26	62.3 % (57.7–66.3)	7.4 % (4.8–10.5)

The ordering is **identical** to the full asymmetric panel. The OpenAI-vs-Anthropic accuracy gap holds (lower bounds 70.8–72.5% vs upper bounds 66.3–67.5%, no overlap). On the matched subset Opus and Sonnet fabrication CIs partially overlap, so we present the *direction* of the Opus-vs-Sonnet finding from the full panel (where Opus's lower bound exceeds Sonnet's upper bound) as the load-bearing claim. The OpenAI-vs-Anthropic fabrication gap is robust on the matched cut.

4.4 Summary vs analysis

Splitting by task surfaces a structural pattern. **Every model fabricates more on the summary task than on the analysis task.**

Model	Summary acc	Analysis acc	Summary fab	Analysis fab
GPT-5.5 Pro	73.9%	74.6%	8.2%	2.3%
OpenAI o3	69.6%	73.7%	7.4%	2.9%
Claude Opus 4.7	60.3%	59.7%	11.0%	10.2%
Claude Sonnet 4.6	58.9%	60.5%	11.1%	5.0%

The interpretation is sharp:

- **Summaries** demand restatement of specific numbers. Models drift on numbers (high `fail`) and occasionally invent figures (high `fab`). Fabrication is concentrated where the model is *trying to be precise* — exactly where it should be most defended.
- **Analyses** allow inference. Models that lean inferential get rewarded on `pass + minor_drift` because their interpretive sentences anchor cleanly to source narrative; their *numeric* fabrication drops because they make fewer specific numeric claims.

A compliance officer should worry more about a one-page summary than a five-page analysis from the same model — the precise opposite of the intuitive prior.

4.5 Failure-mode signature per model

Each model has a characteristic fingerprint, visible in raw verdict counts:

Verdict	GPT-5.5 Pro	OpenAI o3	Opus 4.7	Sonnet 4.6
pass	377	915	855	954
derived_pass	110	272	278	293
minor_drift	64	270	199	167
inference_unanchored	15	93	60	44
inference_unverified	77	156	248	226
fail	61	224	345	487
fabrication	38	108	235	201

- **GPT-5.5 Pro** — when it errs, it errs slightly: small numeric drift, occasional unverified inference. Fabrication is rare (38 / 742 ≈ 5.1%) and well below the Anthropic models. *The right default for a compliance summary.*
- **o3** — middle of the pack. Larger volume of `fail` (224 / 2,038 ≈ 11.0%) than `fabrication` (108 / 2,038 ≈ 5.3%). *Errors are correctable on review.*

- **Opus 4.7** — fabrication is the dominant failure mode (235 / 2,220 ≈ 10.6%). Errors are *invented*, not corrupted — the harder class to catch by hand. *Wrong default for material that needs to be defended.*
- **Sonnet 4.6** — fail-dominant (487 / 2,372 ≈ 20.5%). Misstates source material rather than inventing it. Easier to catch on review than Opus, but more frequent.

This is the kind of structured signal a buyer needs in order to choose a model for a use case, and exactly the signal that no LLM-as-a-judge approach can produce reliably (because the judge shares the same priors).

4.6 Determinism check

The auditor contains no generative model and no sampling, so identical inputs should produce identical verdicts. We verified this empirically by re-auditing 12 randomly sampled cells (3 per model) and comparing verdict counts to the originals:

- **11 / 12 cells produced bit-identical verdict counts.**
- The single divergent cell (`JPM_o3_summary`) returned 37 claims on replay vs 39 originally — a 2-claim drift in the upstream claim-extraction step (sentence segmentation on a borderline punctuation case), with the resulting verdict shifts confined to that delta. The verdict assignment for every claim that *was* extracted in both runs was identical.

The honest formulation is: **the verdict pipeline is deterministic; claim extraction is deterministic to within a small (<2 claims per ~40) edge-case variance from sentence segmentation.** No LLM judge can offer even this level of reproducibility — a reviewer cannot rerun a judge model and get the same verdicts. The replay script (`scripts/determinism_check.py`) is included in the repo so any reader can reproduce the test.

4.7 What the auditor cannot do

We are explicit about the system's limits:

- It cannot verify *truth* in the world — only consistency with a supplied source. If the source is wrong, the audit will pass a wrong-but-restated claim.
- It cannot fully judge inferences. By design we mark unanchored inference as `inference_unanchored` rather than `fail`. A human reviewer adjudicates inference; the auditor flags it for them.
- It is sensitive to source quality. Tables that survive PDF extraction badly, transcripts with missing punctuation, and OCR'd 8-Ks all degrade anchor quality. Ingestion improvements compound.

These are honest, scoped limits. They are also exactly the limits a compliance buyer expects from a deterministic system; opaque LLM judges cannot enumerate their own limits.

---

5. Productisation

The implementation in this repository is the working product, not a slide:

- **Unified CLI.** `armour audit | generate | batch | serve | report` covers the full workflow from a single source / output pair to a parallel pipeline across an entire dataset directory. The full v2 evaluation (240 jobs queued, 203 completed across two top-up rounds before OpenAI quota was exhausted) ran end-to-end in **<35 minutes of wall time** on a laptop, with the deterministic auditor adding zero token cost

beyond generation.

- **HTTP API.** `armour serve` exposes `/audit`, `/audit/report`, `/generate`, `/verdicts`, `/models`, with OpenAPI docs at `/docs`. Designed to embed in an existing compliance workflow rather than replace it.
- **HTML report.** A self-contained, asset-free HTML document showing per-claim verdicts, source anchors, and domain scores. Readable by a non-technical reviewer; shareable as a single file.
- **Parallel batch runner.** Per-provider rate-limited semaphores, retry / backoff with exponential jitter, process-pool audits, streaming JSONL aggregate so partial runs are not lost. Skip-if-exists so a partial run resumes without recomputation.
- **Dataset harvester.** `harvest_press_releases.py` fetches the most recent 8-K Item 2.02 EX-99.1 exhibit per ticker from SEC EDGAR. The 30-ticker dataset in this paper was assembled in under two minutes.

5.1 Target market

Three underserved segments:

1. **Boutique asset managers ($500m–$10bn AUM).** Use AI-generated research routinely. Have no internal model-risk function. Cannot defend an LLM-only workflow to a regulator or LP. Will pay for a defensible audit trail.
2. **Independent research firms.** Sell research on subscription. Reputation risk on every note. A per-note audit ledger is a *credential*, not a cost.
3. **Mid-tier bank compliance.** Cannot match Goldman / JPM in-house ML capability but face the same regulatory pressure. A vendor-supplied deterministic audit layer is precisely the kind of artefact the SR 11-7 / model-risk world is built to consume.

The large incumbents (Bloomberg, FactSet, S&P) will eventually build something like this. They will build it badly first, and they will build it to defend their data moat — not to give you receipts. There is a window.

5.2 Why now

- LLM use is past the experimentation phase in finance. The next 24 months are about *governance*, not capability.
- The SR 11-7-style frame for model risk applies to LLMs essentially unchanged. Any vendor providing AI-generated content into a regulated workflow will need an audit layer underneath it, soon.
- "LLM-as-a-judge" is starting to receive academic and industry pushback. The market is open for a structurally different answer.
- The empirical gap between marketed model quality and audited model quality (Section 4.2) is large enough — 10.6% fabrication on a flagship model — that buyers can no longer responsibly accept self-reported quality claims.

---

6. Roadmap

- **Source-form coverage.** Beyond press releases and transcripts: 10-K / 10-Q sections, broker-dealer notes, regulatory filings, central-bank speeches.
- **Cross-document verification.** Audit a *single output* against *multiple sources* (e.g. a sector note against five constituent releases).

- **Numerical reasoning depth.** First-class arithmetic graph — recompute YoY, margin, sequential growth from canonical source values, not just match-by-presence.
- **Entity normalisation.** "Apple Inc.", "Apple", "AAPL", "the Company" all collapse to the same scope unit.
- **Embedding anchor by default.** MiniLM is small and fast; default-on once we measure precision/recall against the token-overlap baseline on a labelled set.
- **Reviewer console.** Lightweight web UI on top of the API for compliance reviewers to triage, comment, and sign-off claim-by-claim.
- **Per-token audit cost reporting.** Pair every audit with the LLM token cost that produced the output, so buyers can compare $/defensible-claim across models.

---

7. Conclusion

The right product surface for AI-in-finance over the next two years is not a better generator. It is a defensible verifier — one that a compliance officer can sign and a portfolio manager can cite. Project Armour is that verifier, built on the deliberate constraint that the audit layer contains no generative AI. The constraint is the product: it is exactly what makes the audit defensible.

The v2 evaluation moves the case from anecdote to evidence: across **7,372 audited claims** drawn from **30 sector-diversified large-cap earnings releases**, every frontier model in the panel produces material, reviewable error. The strongest model fabricates on ~5% of claims; the weakest, on ~11%. *Every* one of these errors is currently invisible to the buyer absent a system like Armour. That is the market.

The rest is execution.

Appendix A — Locked prompts

Used for every model in every run. Identical across the panel; not tuned per model.

Summary

> You are a financial analyst reviewing an earnings call transcript. Produce a structured summary grounded entirely in the transcript. Include: (1) exact financial figures reported (with periods), (2) management's stated drivers and commentary, (3) forward guidance as stated, (4) key analyst Q&A themes.

> CRITICAL: Do not infer, interpolate, or add context from outside the transcript. Only state what was explicitly said. Approximately 400–500 words.

Analysis

> You are a financial analyst. Based ONLY on the provided transcript, provide: (1) what the financial trajectory implies about business momentum, (2) assessment of the precision and credibility of stated guidance, (3) key risks/uncertainties explicitly mentioned or directly implied by the data, (4) whether reported results are consistent with management's narrative.

> Base every assertion on specific facts from the transcript. If you infer or interpret, flag it explicitly as interpretation. Approximately 500–600 words.

Appendix B — Reproducing the evaluation

Harvest the dataset:

```
py src\harvest_press_releases.py --tickers AAPL AMZN GOOGL MSFT NVDA META AMD AVGO ` JPM
GS BAC WFC BLK JNJ PFE MRK ABBV UNH TSLA WMT COST HD NKE PG KO BA CAT XOM CVX T `
--output transcripts\dataset
```

Run the panel + audit:

```
$env:PYTHONIOENCODING = "utf-8" py src\cli.py batch ` --dataset transcripts\dataset `
--output batch_results_v2 ` --models gpt55 o3 opus47 claude ` --openai-concurrency 6
--anthropic-concurrency 6 --audit-workers 4
```

Aggregate:

```
py src\analyze.py batch_results_v2
```

Outputs:

- `batch_results_v2/<TICKER>/<TICKER>_<model>_<task>.txt` — raw LLM outputs.
- `batch_results_v2/<TICKER>/<TICKER>_<model>_<task>_audit.json` — per-pair audit JSON.
- `batch_results_v2/batch_results.jsonl` — streaming aggregate, crash-safe.
- `batch_results_v2/full_cross_model_comparison.csv` — every (ticker, model, task) row.
- `batch_results_v2/per_model_summary.{csv,md}` — per-model rollup.
- `batch_results_v2/whitepaper_stats.json` — machine-readable inputs for this paper.

Per-pair HTML reports:

```
Get-ChildItem batch_results_v2 -Recurse -Filter *_audit.json | ForEach-Object { py
src\cli.py report $_.FullName }
```

Appendix C — Coverage caveat

Anthropic models (Sonnet 4.6, Opus 4.7) completed all 30 tickers × 2 tasks = 60 cells each. OpenAI models hit `429 insufficient_quota` and `rate_limit_exceeded` across the initial run and two top-up rounds; o3 completed 55 / 60 cells, GPT-5.5 Pro completed 29 / 60. The 13 tickers with full 4-model coverage are AAPL, ABBV, AMD, AMZN, AVGO, BA, BAC, BLK, CAT, COST, CVX, GOOGL, GS. Per-model statistics in Section 4 are computed over each model's actual cell count; the absolute ordering between OpenAI and Anthropic models is consistent across the 13-ticker full-coverage subset and the full asymmetric set. Re-running the OpenAI models on the residual 17 tickers is a one-line operation once quota is restored.