

PolicyStrata: Testing Policy Drift in LLM Data Agents

Zachary Roth
Raintree Technology
USA
zacharyroth@pm.me

ABSTRACT

PolicyStrata is a deterministic regression-testing and stack-audit tool for governed LLM data-agent stacks. It detects cross-layer policy drift among model-visible manifests, grammars, semantic-plan validators, SQL compilers, database policy and row-level security, lineage, and release filters. Given canonical authorization, semantic, and release specifications, PolicyStrata generates principals, requests, semantic plans, database states, version vectors, and lowerings; checks responsibility-scoped contracts; and emits minimized witnesses that identify the first violated transition. Its doctor mode audits what a real stack has wired: SQL traces, dbt semantic files, PostgreSQL schema/RLS metadata, privacy policies, terms of service, data-processing and internal-policy documents, prompt/tool manifests, source maps, release coverage, and CI gates. The demonstration runs the reproducible artifact path, inspects JSONL traces and witnesses, shows an audit report with remediation todos, and compares PolicyStrata with grammar-only, validator-only, SQL-policy, DB-only, release-only, final-answer, and defense-in-depth baselines.

KEYWORDS

LLM data agents, policy testing, authorization, semantic drift, text-to-SQL, row-level security, regression testing

1 PROBLEM

LLM data agents translate natural-language requests into semantic plans, SQL queries, and analytical answers. In deployed systems, authorization and business-semantics policies are replicated across independently maintained surfaces: model-visible manifests, grammars, semantic-plan validators, SQL compilers, database roles, row-level security policies, lineage trackers, caches, logs, and release filters. A component can pass its own tests while the composed stack violates tenant scope, metric semantics, lineage, or release policy.

For example, a support analyst may be authorized to see aggregate ticket counts for tenant A, but not tenant B or raw customer emails. After a policy update, the validator may run version 7, the manifest may still expose a version-6 alias, and the compiler may still lower through a version-5 tenant key. A request such as “last month’s escalations by region” can then be accepted, compiled with stale context, and either contained by database RLS or released. Testing only final answers, SQL syntax, or grammar membership does not localize where the policy obligation failed.

2 USERS AND USE CASES

PolicyStrata is intended for engineers and researchers building governed data agents, BI copilots, embedded analytics assistants, product analytics agents, and internal data assistants. The primary

users are teams that maintain manifests, semantic layers, validators, SQL compilers, database policies, and release filters independently and need CI evidence that policy obligations survive translation across those layers.

Concrete use cases include tenant-isolation regression after schema migrations, semantic-model migration testing for revenue and support metrics, release-boundary checks before sending data to external models, privacy and terms-of-service obligation coverage checks, and artifact evaluation for data-agent benchmark papers.

For the single-blind tool demonstration, we also include a BetterOff finance-assistant integration fixture over real application surfaces with synthetic fixture data: policy and terms documents, a prompt manifest, source map, semantic model, SQL/tool traces, Docker PostgreSQL RLS fixture, and CI workflow wired end to end.

3 TOOL DESIGN

PolicyStrata treats governed data-agent safety as a cross-layer conformance problem. It separates policy into authorization, semantic, and release specifications, then assigns responsibility-scoped obligations to each surface:

- manifests and grammars should not expose unauthorized reachable operations;
- validators should accept authorized objects inside the declared support envelope and reject unauthorized objects;
- compilers should preserve principal, tenant, purpose, policy version, lineage, and business semantics;
- database policy should contain unauthorized executable operations under the runtime role;
- release filters should release only allowed result-lineage pairs.

The tool generates principals, requests, semantic plans, database states, implementation version vectors, and lowerings. It then checks each layer with independent authorization, semantic, and release interpreters rather than reusing the SQL compiler path. When a contract fails, PolicyStrata emits a minimized witness containing the principal, request, database state, version vector, runtime object, lowered object, lineage, observed result, first violated transition, and violated obligation.

For deployment readiness, `policystrata doctor --config` reports what is wired and what is missing. It classifies privacy policies, terms of service, data-processing agreements, security policies, retention policies, and internal access-control documents into obligation signals; parses JSON/YAML prompt manifests to compare exposed metrics and dimensions against the canonical policy; statically introspects PostgreSQL schema SQL for tables, grants, RLS policies, tenant columns, sensitive columns, views, and indexes; and emits remediation todos with an owner, expected files, expected tests, and a CI gate command. This audit is deterministic

coverage accounting, not legal review and not a production authorization guarantee.

4 DEMONSTRATION PLAN

The demonstration is organized around one cross-tenant analytics failure and the deterministic reproduction path.

- (1) Show a support-analytics principal scoped to tenant A.
- (2) Trigger a stale compiler/tenant-key mutation.
- (3) Inspect the minimized witness: principal, request, version vector, semantic plan, SQL, lineage, result, first violated transition, and obligation.
- (4) Show whether database containment blocked the unauthorized behavior.
- (5) Run `policystrata doctor --config` on a scanner fixture and inspect privacy and terms-of-service document coverage, prompt-manifest drift, source-map coverage, release coverage, and concrete remediation todos.
- (6) Switch to the BetterOff integration fixture and show the same doctor and scan gates running against real application wiring with synthetic fixture data.
- (7) Compare point baselines and ablations.
- (8) Show clean controls for intentional underexposure, stricter release suppression, and authorized-but-unsupported requests.

The main command sequence is:

```
uv run pytest
uv run ruff check .
uv run mypy src
POLICYSTRATA_RUN_ROOT=/tmp/policystrata-final-submit \
./scripts/reproduce-final.sh
uv run policystrata doctor --config \
  examples/postgres_dbt/policystrata_real_db_clean.yaml \
  --format markdown --out runs/doctor.md
```

The final script writes JSONL traces, summaries, minimized witnesses, freeze manifests, baseline reports, ablations, export-adaptor outputs, and an artifact report. The doctor command writes a stack-readiness report and remediation checklist. Neither path requires an LLM API key or host `psql`. Optional Docker services exercise PostgreSQL/RLS and ClickHouse smoke paths.

5 VALIDATION SNAPSHOT

The artifact includes DataPolicyDriftBench, a deterministic benchmark with three analytics domains: B2B support analytics, household/advisor finance analytics, and ClickHouse-style product analytics. The suites include seeded cases, generated cases, detector-frozen held-out cases, and clean controls.

On controlled deterministic non-equivalent mutant suites, PolicyStrata reports witnesses for 1720/1720 injected faults covered by the implemented operators and fixtures, with 0 false positives on 80 clean controls. Baseline comparators are evaluated over the same traces. Point baselines catch substantially fewer failures because they observe only one layer or artifact class; the strongest defense-in-depth stack catches 1561/1720 failures. These are deterministic artifact-suite results, not a claim of recall on unknown production faults and not an authorization-boundary claim.

Table 1: Representative baseline comparison over the same 1720 non-clean traces.

Comparator	Failures caught
Grammar only	121/1720
Semantic validator only	573/1720
SQL AST policy checker	1043/1720
DB policy only	326/1720
Release filter only	364/1720
Defense-in-depth stack	1561/1720

As downstream integration smoke evidence, the BetterOff fixture currently reports CI gate pass, 0 findings, `ci-gate-ready`, 7 imported-trace checks, and 4 real database checks. This is real-application wiring evidence with synthetic fixture data for the tool demonstration, not production customer-data evidence and not part of the 1720-case deterministic benchmark score.

6 COMPARISON

Text-to-SQL benchmarks such as Spider and BIRD evaluate query correctness [3, 7], and constrained decoding systems such as PISCARD reduce malformed outputs [6]. Distilled test-suite evaluation and bounded SQL-equivalence checking show why generated databases and semantic comparison matter for query behavior [2, 8]. Access-control mutation testing and policy-difference analysis motivate fault models for policy adequacy [1, 4]. PostgreSQL row-level security provides an important containment layer, but its bypass and owner behavior make executable-role testing necessary [5]. These techniques are useful reliability and enforcement layers, but they do not test whether authorization, semantic, lineage, and release obligations survive across independently versioned data-agent surfaces. PolicyStrata instead tests the conformance relationships among those layers and emits a database-backed witness for the first violated transition. This makes it complementary to policy enforcement, SQL equivalence checking, and database row-level security rather than a replacement for them.

7 TOOL AVAILABILITY

Latest tool version:

- Tagged tool snapshot: <https://github.com/rainforest-technology/policystrata/tree/v0.1.5>
- Public paper: <https://rainforest.technology/papers/PolicyStrata.pdf>
- Documentation: README, evaluator guide, expected-results document, methodology notes, CLI help, deterministic reproduction script, and test suite.

The current pushed source tags are:

- `v0.1.5`
- `policystrata-doctor-ci-gates-2026-06-26`
- `policystrata-doctor-audit-2026-06-26`
- `policystrata-paper-layout-fix-2026-06-26`
- `policystrata-website-layout-fix-2026-06-26`

Demonstration video: `TODO-YOUTUBE-URL`. Archived version: the tagged GitHub source snapshot above is the frozen public

source reference for the tool demonstration submission; add a Zenodo DOI if a DOI-bearing archive is preferred.

REFERENCES

- [1] Kathi Fisler, Shriram Krishnamurthi, Leo A. Meyerovich, and Michael Carl Tschantz. 2005. Verification and Change-Impact Analysis of Access-Control Policies. In *Proceedings of the 27th International Conference on Software Engineering*. 196–205.
- [2] Yang He, Pinhan Zhao, Xinyu Wang, and Yuepeng Wang. 2024. VeriEQL: Bounded Equivalence Verification for Complex SQL Queries with Integrity Constraints. *Proceedings of the ACM on Programming Languages* 8, OOPSLA2 (2024). Also available as arXiv:2403.03193.
- [3] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Min Zhou, Fei Zhang, Guoliang Li, Kevin Chen-Chuan Chang, Xuanjing Huang, Reynold Cheng, and Yongbin Li. 2023. BIRD: A Big Bench for Large-Scale Database Grounded Text-to-SQL Evaluation. *arXiv preprint arXiv:2305.03111* (2023).
- [4] Evan Martin and Tao Xie. 2007. A Fault Model and Mutation Testing of Access Control Policies. In *Proceedings of the 16th International Conference on World Wide Web*. 667–676.
- [5] PostgreSQL Global Development Group. 2026. PostgreSQL Documentation: Row Security Policies. <https://www.postgresql.org/docs/current/ddl-rowsecurity.html>. Accessed 2026-06-24.
- [6] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. *arXiv preprint arXiv:2109.05093* (2021).
- [7] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- [8] Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 396–411. <https://doi.org/10.18653/v1/2020.emnlp-main.29>