

How We Know Pre-Action Gates Work: An Evaluation Framework for Infrastructure Firewalls on AI Coding Agents

Igor Ganapolsky
iganapolsky@gmail.com
ThumbGate Project
<https://thumbgate.ai>

<https://huggingface.co/spaces/IgorGanapolsky/ThumbGate>

August 2026

Abstract

AI coding agents repeatedly commit the same high-cost failures—force-pushes to protected branches, secret egress, unreviewed outbound mail, and other tool actions that should never execute without policy. Pre-action enforcement intercepts tool calls before execution (e.g., Pre-ToolUse hooks) and decides allow, warn, or deny from deterministic gates, lessons, and human-review routes. This paper presents a practical evaluation framework for such infrastructure firewalls, instantiated by ThumbGate: an open-source, local-first system that promotes operator feedback into prevention rules and hard blocks. We define seven proof dimensions—golden datasets, offline regression, tool-call correctness, latency structure, cost avoidance, human review, and production monitoring—and map each to committed tests, public scorecards, and re-run commands. The framework requires `unsafeActionRate = 0` on golden suites and treats production incidents as golden candidates. Artifacts and demos are available on GitHub, npm (thumbgate), and Hugging Face Spaces.

1 Introduction

Coding agents with shell, filesystem, and MCP tool access can burn money and trust in a single wrong command. Training-time alignment and post-hoc log review do not stop the bad tool call before it runs. Pre-action gates close that gap: every tool invocation is evaluated against rules derived from prior failures.

ThumbGate implements this loop as: capture thumbs feedback → promote lessons → generate prevention rules → enforce at PreToolUse → measure and feed misses back into goldens. The product claim is not “AI safety in the abstract,” but measurable decision quality under real agent harnesses (Claude Code, Cursor, Codex, OpenCode, MCP).

This paper does not report multi-institution human studies. It specifies how an operator or buyer should know an enforcement layer works, with evidence paths that are public and re-runnable.

2 Threat model and scope

We target local and hybrid agent runtimes that execute developer tools. In scope: force-push, secret egress, financial mutations, outbound messaging hard floors, and supply-chain install patterns.

Out of scope: model-internal jailbreak robustness as the primary control, and claims of customer revenue.

3 System overview

ThumbGate is local-first: gates and lessons live on the operator machine; hosted Pro is optional for sync and dashboard. Enforcement is deterministic on the hot path (pattern and rule match first). Semantic retrieval and LLM judges are diagnostic or offline, not required for hard deny of known-bad patterns.

4 Seven proof dimensions

4.1 Golden evaluation dataset

Committed packs define expected allow/deny/warn outcomes, including ThumbGate Bench, agent-safety evals, prompt and observability suites, and shell golden tests. Production failures promote into offline goldens via behavior monitoring (goldenCandidates).

4.2 Offline regression

Standard verification before merge or release:

```
npm test
npm run test:coverage
npm run prove:adapters
npm run prove:automation
npm run self-heal:check
```

Evidence is recorded in public verification docs and release-confidence checks (Changesets, SemVer, version sync).

4.3 Tool-call correctness

Bench metrics include task success, unsafe action rate (must stay 0), blocked-unsafe rate, capability rate (safe work allowed), false-block rate, and replay stability. Public scorecard: <https://thumbgate.ai/eval-scorecard>. Reproduce: `npm run thumbgate:bench -- --json`.

4.4 Latency

No LLM on the enforcement decision path. Deterministic match first (sub-millisecond class); local semantic fallback only when needed. Routing budgets constrain low-latency paths (e.g., 300 ms class budgets in classifier routing config).

4.5 Cost

Blocks avoid model round-trips; budget ledgers and tokenomics guards track spend; public methodology documents avoided-repeat savings without inventing revenue.

4.6 Human review

High-risk classes (credentials, customer data, regulated work, payments) route to human review. Rubrics require verification evidence. Social and email outbound default to draft-only until a human sends.

4.7 Production monitoring

Deploy health endpoints, self-heal checks, gate statistics, silent-gate canaries, and congruence probes. Monitors must feed failures back into goldens—absence of alerts is not evidence of safety.

5 Buyer audit protocol

In about twenty minutes an independent reviewer can:

1. Clone <https://github.com/IgorGanapolsky/ThumbGate> and run `npm ci`.
2. Run the verification suite and bench JSON export.
3. Probe production `/health` and public scorecard pages.

6 Limitations

This framework measures gate decision quality and operator workflow reliability. It is not a substitute for penetration testing of every integration, nor proof of third-party paid adoption. Default free installs may not hard-block every risky pattern without configuration.

7 Related work

Agent tool use and MCP expand the action surface of LLMs. PreToolUse-style hooks (agent harnesses) provide the interception point. Our contribution is the closed loop from feedback to enforceable gates with an explicit evaluation contract buyers can re-run.

8 Artifacts

- Source: <https://github.com/IgorGanapolsky/ThumbGate>
- Package: `npm thumbgate init`
- White paper (HTML): <https://thumbgate.ai/whitepaper>
- Hugging Face Space: <https://huggingface.co/spaces/IgorGanapolsky/ThumbGate>
- Sample feedback schema dataset: <https://huggingface.co/datasets/IgorGanapolsky/thumbgate-agent-feedback>

9 Conclusion

Pre-action gates for coding agents should be evaluated like safety-critical control software: golden suites with zero unsafe-action rate, offline regression, explicit latency and cost structure, human review for residual risk, and production monitors that regenerate goldens. ThumbGate implements and dogfoods this contract as an open infrastructure firewall.

Acknowledgments

Built for operators of multi-agent coding fleets who need enforcement, not only dashboards.

References

- [1] I. Ganapolsky. ThumbGate documentation and verification evidence. <https://github.com/IgorGanapolsky/ThumbGate>, 2026.
- [2] I. Ganapolsky. ThumbGate Hugging Face Space. <https://huggingface.co/spaces/IgorGanapolsky/ThumbGate>, 2026.