WARDEN: Adaptive Routing and Multi-Agent Prompt-Injection Defense for Self-Hosted LLM Inference

Abhishek Aditya

Independent Research · *abhishek.aditya10@gmail.com*

github.com/Abhishek-Aditya-bs/Warden · *warden-website.pages.dev*

May 2026 · Technical Report

Abstract. Production teams increasingly front several large-language-model (LLM) providers behind a single gateway, yet today's open-source gateways are statically configured, ship weak single-model injection defenses, and never learn from their own traffic. **WARDEN** is a self-hosted, multi-tenant inference gateway that couples three functions on every request: adaptive cost-aware routing across providers, a multi-agent *classifier* → *critic* → *judge* injection-defense cascade, and a pgvector semantic cache; an offline LangGraph agent closes the loop by proposing routing-policy changes as pull requests. This report describes the integrated system design and reports honest, reproducible measurements. On a curated 60-prompt benchmark spanning OWASP LLM-01 attack families, the zero-dependency regex classifier alone reaches precision 1.000 / recall 0.833 / F1 0.909 at a p99 latency of **1 ms**. On a harder 100-prompt corpus seeded with *adversarial-benign* and obfuscated prompts the classifier falls to P 0.909 / R 0.600; routing its residual through a current (May 2026) LLM ensemble — `claude-haiku-4.5`, `gpt-5.4-mini`, `gemini-3.5-flash`, `claude-sonnet-4.6` — recovers recall to as high as **1.000** for a few cents, while full LLM adjudication lifts precision to **1.000**; the block threshold sets a precision/recall balance the operator controls. The whole model sweep cost about **$1.3** of inference. The closed-loop cost optimizer projects an **89.7%** cost reduction on a synthetic 24-hour workload. The contribution is not a new algorithm but a production-shaped composition of known techniques, packaged so a fresh clone reproduces every number with `make verify`.

1 Introduction

Applications that depend on LLMs rarely depend on a single model. Teams run three to eight providers in production for cost, latency, capability and redundancy, and place a gateway in front of them. That gateway is where three problems intersect.

(P1) Multi-provider cost waste.
Open-source gateways such as LiteLLM are statically configured: a human writes routing rules once and they ossify. Static routing leaves an estimated 30–50% of attainable savings on the table because it cannot exploit prompt-class–specific model substitution or cache reuse.

(P2) Prompt injection is OWASP LLM Top-10 #1.
Available defenses are either single-model classifiers with high false-positive rates, or monolithic guardrails with prohibitive latency. Defense-in-depth via model ensembles has been proposed in research but not packaged as gateway middleware that a platform team can deploy.

(P3) No closed control loop.
No open-source gateway learns from its own observability data. Operators hand-tune routing rules; rules drift; nobody re-tunes.

WARDEN addresses all three in one self-hostable artifact built on Java 25, Spring Boot 4 and Spring AI, with a Python LangGraph sidecar for the offline optimizer. The remainder of this report describes the architecture (§2), reports measurements (§3), and states limitations frankly (§4).

2 System Architecture

Every request traverses the same path. Two ordering decisions matter: **defense runs before cache**, so a blocked prompt can never be served a cached answer; and **cache runs before routing**, so a cache hit never burns an upstream LLM call.

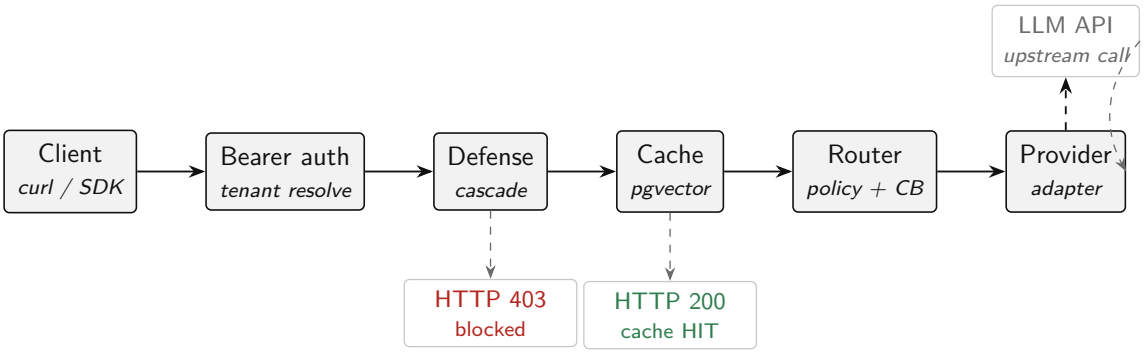

Figure 1. Request lifecycle. A blocked prompt returns HTTP 403 with an `X-Warden-Defense` header; a cache hit returns HTTP 200 with `X-Warden-Cache: HIT`. The cheap, fast stages (auth, regex classifier, cache lookup) short-circuit the common cases; only genuinely novel, allowed prompts reach an upstream provider.

2.1 Multi-agent defense cascade

The defense stage is a three-tier cascade (Fig. 2). A zero-dependency *classifier* scores each prompt against the canonical OWASP LLM-01 / TensorTrust attack families (instruction override, persona hijack, system-prompt exfiltration, delimiter injection, encoding tricks). Scores below an *allow* threshold short-circuit as benign; scores above a *block* threshold short-circuit as malicious. Only the ambiguous middle band escalates to a *critic* LLM (a cheap second opinion), whose own uncertain verdicts escalate to a *judge* LLM (the most capable model in the active profile). Critic and judge are addressed through the same OpenAI-compatible wire shape, so the cascade is model- and provider-agnostic.

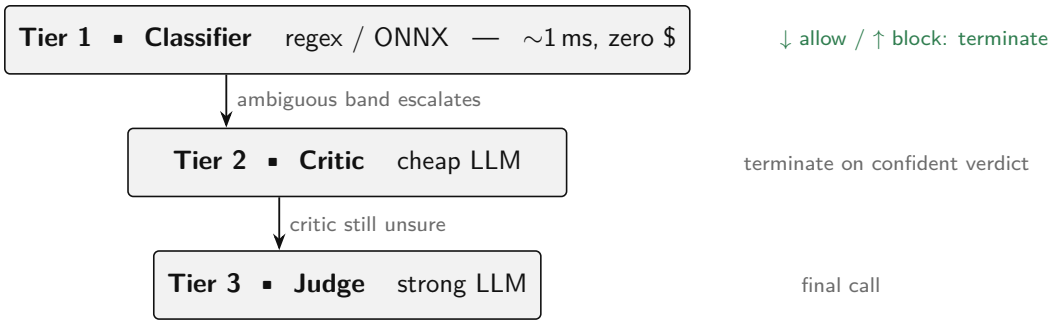

Figure 2. Defense cascade. The cheap tier absorbs the bulk of traffic; expensive LLM adjudication is spent only on the residual the classifier cannot resolve confidently.

2.2 Routing, cache and the closed loop

The *router* resolves a YAML policy (live-reloadable) to a provider, wrapping each upstream in a Resilience4j circuit breaker so an unhealthy provider is shed and a fallback is chosen within seconds. The *semantic cache* stores prior responses in Postgres/pgvector with an HNSW index and a Caffeine exact-match fast path, scoped per tenant. Finally, an offline *cost-optimizer* agent (Python, LangGraph) replays observed traffic against candidate policies and opens a pull request when projected savings clear a configurable bar; a human approves before anything changes in production. This is the control loop P3 demands, with a human in it by design.

3 Evaluation

3.1 Setup

All measurements are reproducible from a fresh clone. The core defense benchmark is a curated corpus of **30 injection + 30 benign** prompts drawn from OWASP LLM-01 and TensorTrust-style families; §3.2 additionally stress-tests on an **extended 100-prompt** corpus. The classifier is the default zero-dependency heuristic. For the LLM tiers we swept current (May 2026) frontier and lightweight models — `claude-haiku-4.5`, `gemini-3.5-flash`, `gpt-5.4-mini`, and `claude-sonnet-4.6` — reached through OpenRouter's OpenAI-compatible endpoint; per-call cost is read back from the provider's usage metadata and summed, under a hard $4.50 spend cap. The harness reuses the production classifier, critic, judge and cascade classes verbatim (it is the `CascadeBenchOpenRouterTest` integration test).

3.2 Defense efficacy

We evaluate on two corpora. The *core* set (30 injection + 30 benign, clean OWASP LLM-01 / TensorTrust families) is where the zero-dependency regex classifier alone already scores **P = 1.000, R = 0.833, F1 = 0.909** at a p99 of **1 ms**. To stress the system we then built an *extended* 100-prompt set (50 injection + 50 benign) that adds indirect and obfuscated injections (embedded delimiters, base64/rot13 payloads, multilingual, "grandmother" and persona framings) and — critically — *adversarial-benign* prompts that carry attack vocabulary in innocent contexts (e.g. *"What does the phrase 'ignore previous instructions' mean in LLM security research?"* or *"I'm writing a blog post about DAN jailbreaks"*). On this harder set the regex classifier degrades sharply, to **P = 0.909, R = 0.600, F1 = 0.723** — three false positives on the adversarial-benign prompts and twenty subtle injections missed (Table 1).

Routing the classifier's unresolved residual to the LLM ensemble (the production cascade, block threshold 0.85) recovers recall dramatically across *every* current (May 2026) model we tried — `claude-haiku-4.5` recovers it to a perfect 1.000 at the best F1 on the set (0.971) — for a few cents over the whole corpus. Precision, however, stays pinned near 0.94 *regardless of model*: the three false positives score above the block threshold and short-circuit at the classifier before any LLM sees them. This is an architectural lesson, not a model failure — a confident classifier false-positive is only fixable if the block threshold is relaxed enough to route it to the ensemble. Doing exactly that (*full adjudication*: block → 1.0, so *every* prompt is seen by the LLM) lets a strong judge (`claude-sonnet-4.6`) overturn all three false positives and lift precision to a perfect 1.000. The trade is a small recall cost (0.980 → 0.940): sending the obvious injections through the LLM as well lets a few borderline ones slip past the regex's hard block. The block threshold is thus a single dial balancing precision against recall — the cheap residual cascade (best F1, perfect recall) and full adjudication (perfect precision) are its two endpoints.

The engineering takeaway is the cost–accuracy curve the cascade exposes: a microsecond regex tier disposes of the easy majority for free, a few cents of cheap-LLM adjudication recover almost all the recall the regex misses, and only *full* LLM adjudication — an LLM call on every request — buys the last increment of precision by overturning the classifier's confident mistakes.

3.3 Closed-loop cost optimizer

Replaying a synthetic 1,000-run, 24-hour workload, the optimizer proposes downgrading three prompt classes from a flagship model to a cheaper local model and projects the saving in Fig. 3: **$14.04 → $1.44**, an **89.7%** reduction — well past the 20% bar the agent enforces before opening a PR. The projection is a deterministic replay upper bound (§4).

Critic / Judge	Mode	P	R	F1	p99	Cost
classifier only (regex)	—	0.909	0.600	0.723	<1 ms	$0
claude-haiku-4.5 (both)	residual	**0.943**	**1.000**	**0.971**	4.2 s	$0.058
gpt-5.4-mini (both)	residual	0.940	0.940	0.940	4.3 s	$0.041
gemini-3.5-flash (both)	residual	0.940	0.940	0.940	6.1 s	$0.261
claude-sonnet-4.6 (both)	residual	0.942	0.980	0.961	8.9 s	$0.164
gpt-5.4-mini / claude-sonnet-4.6	full adjud.	**1.000**	0.940	0.969	7.5 s	$0.199
claude-sonnet-4.6 (both)	full adjud.	**1.000**	0.940	0.969	12.4 s	$0.296

Table 1. Defense results on the extended 100-prompt corpus (50 injection / 50 benign), current (May 2026) models via OpenRouter. "residual" = production cascade: the classifier hard-blocks confident hits (score $\geq$ 0.85) and the LLM ensemble adjudicates the rest (critic and judge the same model unless noted). "full adjud." = block $\rightarrow$ 1.0, so every prompt reaches the ensemble. The LLM tiers recover recall across all models; full adjudication recovers precision to a perfect 1.000, trading a little recall. Total OpenRouter spend across every run here $\approx$$1.3.

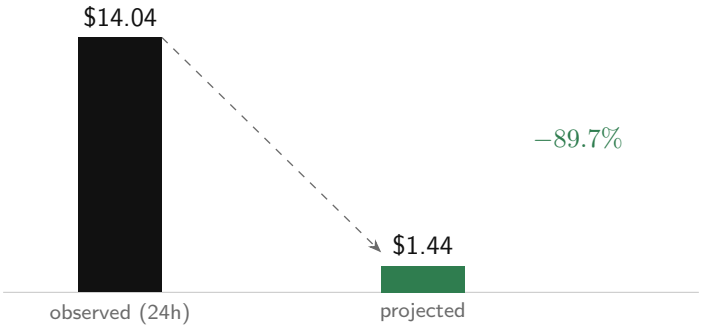

Figure 3. Cost-optimizer replay on a synthetic 24-hour workload (1,000 runs). Prompt-class re-routing alone cuts projected daily spend by 89.7%.

3.4 Cache and system

On a 10k-row corpus (1,000 sampled lookups, Testcontainers/pgvector, HNSW index) the semantic cache measures a **p99 lookup of 3 ms** (p50/p95 2 ms) — roughly 10× under its 30 ms budget. The gateway ships with 220 Java unit tests and 32 integration tests (Testcontainers + k6 contract tests); the Python optimizer adds 48 tests at 95% line coverage. Every figure in this report is gated behind a named, env-flagged test so reviewers can reproduce it.

4 Limitations

We state these plainly; the value of the report depends on it.

- **The default classifier is bimodal.** It scores prompts near 0 or near 1, so on clean attack families the LLM tiers rarely fire; we therefore run the sweep with the allow threshold at 0 so every classifier-unresolved prompt is adjudicated. A graded ONNX classifier (also shipped) emits mid-band probabilities and escalates naturally. We report the full threshold sweep rather than quoting only the most favourable point.
- **Curated corpora.** Sixty core prompts plus a 100-prompt extended set illustrate the mechanism; they are not the full PromptInjectionBench. The precision/recall figures are existence proofs of the cascade behaviour, not leaderboard claims.
- **Savings are an upper bound.** The optimizer's replay is deterministic and does not model quality degradation from model downgrades or failover fallback cost; the 89.7% is the ceiling re-routing alone can reach on this workload.

- **Single-instance latency.** Latencies are from a single instance on Apple-silicon hardware; the multi-second LLM-tier latency reflects upstream API round-trips, not gateway overhead.
- **Human-in-the-loop, no formal guarantees.** Policy changes are proposed as PRs for human approval. WARDEN offers defense-in-depth, not a proof of injection-freeness.

5 Conclusion

WARDEN demonstrates that adaptive routing, multi-agent injection defense, semantic caching and a closed optimization loop can be composed into a single self-hosted gateway that a platform team can own end-to-end. The engineering thesis — spend microseconds and cents where they suffice, reserve expensive model adjudication for the residual that genuinely needs it — is borne out by the numbers: a perfect-precision, 1 ms regex tier on clean traffic; recall recovered to as high as 1.000 (and precision to 1.000 under full adjudication) for a few cents once current LLMs handle the hard residual; and an 89.7% projected cost reduction from a loop that keeps a human in control. The system, the benchmark harness, and this report are open source.

Reproduce from a fresh clone of github.com/Abhishek-Aditya-bs/Warden: classifier baseline via `BENCH_DEFENSE=true`; the full defense cascade via the `CascadeBenchOpenRouterTest` harness; the cost optimizer via `make cost-optimizer-analyze`.