Representation Rerouting for Agentic Safety

Internal Defenses Against Prompt Injection via LoRA Circuit Breakers and Triplet Loss

Internal Research Report · February 2026 · Base model: `Llama-3.1-8B-Instruct`

Artifacts: huggingface.co/memo-ozdincer/rrfa-runs · huggingface.co/datasets/memo-ozdincer/rrfa-data

Mehmet Ozdincer, Samuel Simko, Zhijing Jin | Jinesis AI Research Group, Vector Institute

Abstract. LLM agents with tool-calling capabilities are vulnerable to prompt injection attacks that hijack control flow via adversarial content in retrieved data. We present **Representation Rerouting for Agentic Safety (RRFA)**, training LoRA adapters to make harmful internal representations orthogonal to benign ones via a triplet loss with configurable loss masking. Evaluated on three benchmarks—Fujitsu B4 (tool-flip), AgentDojo (multi-domain), and LLMail-Inject (email agent)—our best configuration reduces Fujitsu ASR from 83.7% to 8.2% (**75.5 pp**) with **zero regressions**, achieving 5.0% LLMail ASR and 100% behavioral change on AgentDojo. Notably, the defense often *restores correct behavior* rather than merely refusing.

1. Introduction

LLM-based agents increasingly operate in high-stakes environments—emails, financial transactions, databases—where a single compromised tool call causes data exfiltration or unauthorized actions. *Prompt injection* embeds adversarial instructions in retrieved data (emails, search results, tool outputs), deviating the model from user intent.

Existing defenses are either **input-level** (filtering, delimiters) or **output-level** (guardrails, monitors)—both fragile against creative attacks. We pursue **representation-level defense**: training the model's internal geometry so harmful states are automatically rerouted.

We extend Circuit Breakers [1] from text-only safety to **agentic tool-calling**, where harm is an incorrect tool invocation rather than toxic text. This is the first application of representation rerouting to tool-calling agents.

Contributions.

1. A **triplet loss formulation** with configurable distance metrics for agentic circuit breaker training.
2. Systematic study of **five loss mask policies** controlling which tokens receive rerouting supervision.
3. Evaluation on **three diverse benchmarks** with distinct threat models and injection semantics.
4. A **complete reproducible pipeline** (ETL → generation → masking → training → evaluation).

2. Background

2.1 Prompt Injection in Agents

Injections are *direct* (in user input) or *indirect* (in retrieved data [2]). In agents, the critical outcome is a *tool-flip*: the injection causes a wrong or malicious tool call (e.g., `send_email` instead of refusing), enabling exfiltration.

2.2 Circuit Breakers

The CB framework [1] trains adapters making harmful representations orthogonal to benign ones. For model θ with frozen reference θ_0:

Original CB Loss [1]

$$\mathcal{L}_{\text{rr}} = \frac{1}{|L|}\sum_{l\in L}\frac{1}{T}\sum_{t=1}^{T}\text{ReLU}\big(\cos(\mathbf{h}_\theta^{(l,t)},\mathbf{h}_{\theta_0}^{(l,t)})\big) \tag{1}$$

$$\mathcal{L}_{\text{ret}} = \frac{1}{|L|}\sum_{l\in L}\frac{1}{T}\sum_{t=1}^{T}\|\mathbf{h}_\theta^{(l,t)} - \mathbf{h}_{\theta_0}^{(l,t)}\|_2 \tag{2}$$

$$\mathcal{L}_{\text{total}} = \alpha(t)\cdot\mathcal{L}_{\text{rr}}(D_s) + \mathcal{L}_{\text{ret}}(D_r) \tag{3}$$

where L = target layers, T = sequence length, $\alpha(t)$ decays linearly. Eq. (1) penalizes positive cosine similarity on harmful samples; Eq. (2) preserves benign behavior via L2 anchoring.

3. Method

3.1 Harm Definition

Given agent with tools $\mathcal{T}$, query q, injected context c:

$$\text{HARM}(q,c) := \big(t_{\text{obs}}(q\oplus c)\neq t_{\text{exp}}(q)\big)\wedge \text{inj}(c)$$

Binary, deterministic, no LLM judge required.

3.2 Paired Data Generation

D_s **(Harmful).** Model run with coercing prompt ($T=0.7$); only attack-succeeding samples retained. AgentDojo: traces where `security==False`.

D_r **(Benign twin).** Same context, injection stripped, defensive prompt ($T=0.3$). Teaches what the model *should* have done.

3.3 Triplet Loss Formulation

We extend CB with a triplet loss structuring the representation space. Let $\bar{\mathbf{z}}_h$ be the batch harmful centroid, $d(\cdot,\cdot)$ a configurable distance:

Triplet Full Loss

$$\mathcal{L}_b = \mathrm{ReLU}\big(d(\mathbf{h}^b_{\theta_0}, \mathbf{h}^b_\theta) - d(\mathbf{h}^b_\theta, \bar{\mathbf{z}}_h) + m_b\big) \quad (4)$$

$$\mathcal{L}_h = \mathrm{ReLU}\big(d(\mathbf{h}^h_\theta, \bar{\mathbf{z}}_h) - d(\mathbf{h}^h_\theta, \mathbf{h}^h_{\theta_0}) + m_h\big) \quad (5)$$

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}\big(p_\theta(\cdot|x^b) \| p_{\theta_0}(\cdot|x^b)\big) \quad (6)$$

$$\mathcal{L}_{\mathrm{total}} = \alpha_b \cdot \mathcal{L}_b + \beta_h \cdot \mathcal{L}_h + \gamma \cdot \mathcal{L}_{\mathrm{KL}} \quad (7)$$

m_b, m_h: margins. $\alpha_b, \beta_h, \gamma$: weighting coefficients.

Distance functions are configurable per-term: $d_{\mathrm{L2}} = \|\mathbf{a} - \mathbf{b}\|_2$, $d_{\cos} = 1 - \cos(\mathbf{a}, \mathbf{b})$, $d_{\mathrm{mix}} = w_1 d_{\mathrm{L2}} + w_2 d_{\cos}$. All experiments use d_{mix} ($w_1 = w_2 = 0.5$).

Intuition. Eq. (4): benign reps stay closer to frozen than to harmful centroid. Eq. (5): harmful reps cluster near centroid and away from frozen. Eq. (6): output distribution fidelity on benign inputs.

3.4 Loss Mask Policies

The mask $\mathbf{m} \in \{0,1\}^T$ determines which tokens receive rerouting loss:

Policy	Tokens Receiving Loss
`assistant_only`	All assistant turn tokens
`asst_and_tool`	Assistant + tool call params
`cb_full_seq`	**Entire sequence** (incl. injection)
`tool_calls_only`	`<|python_tag|>{..}<|eom_id|>`
`completion_only`	Final assistant completion

Table 1: Loss mask policies (LMPs).

3.5 Architecture

LoRA. $r = 16$, $\alpha_{\mathrm{LoRA}} = 32$, dropout 0.05 on all projections (`q/k/v/o/gate/up/down_proj`).

Single-model trick. Instead of a separate frozen model (2× VRAM), we reuse the same model: adapters enabled → θ; adapters disabled via `disable_adapter()` → θ_0. Halves memory; enables `MAX_SEQ=4096` on 80 GB H100.

Training. Batch 1, grad_accum 4, AdamW 5×10^{-5}, 200 steps/config, warmup 20, linear α decay over 2×steps.

4. Datasets

	Fujitsu B4	AgentDojo	LLMail
# Records	13K+	194	∼60
Inj. location	User query	Tool responses	Email body
Harm defn.	Wrong tool	Unauth. action	Any tool call
Correct bhvr.	Expected tool	Task safely	Refuse/no call
Trace type	Skeleton (B1)	Complete (B2)	Skeleton (B1)
Generation	DS/DR via vLLM	Split existing	DS/DR via vLLM

Table 2: Dataset comparison. LLMail has *inverted* semantics: correct behavior is inaction.

Fujitsu B4. Tool-flip attacks: injection flips `retrieve_multimodal_docs` → `search_web`. Each record has `benign_query`, `malicious_injection`, `expected_tool`, `simulated_tool`.

AgentDojo [2]. Multi-domain (banking, workspace, travel). Injections in `<INFORMATION>` tags inside tool responses. 194 traces from Claude/GPT-4o/Gemini.

LLMail-Inject. Email agent; injection tries to elicit `send_email`. Success = any tool call. Dedicated `evaluate_llmail_attack()` and `evaluate_llmail_usefulness()` metrics.

5. Pipeline

Six stages, fully automated via SLURM:

ETL_A → Gen → ETL_B → Train → Eval

ETL_A: raw → `trace_v1`. **Gen**: skeleton → DS/DR via vLLM. **ETL_B**: render + LMP mask. **Train**: `CircuitBreakerTrainer`. **Eval**: ASR + capability.

6. Experimental Setup

Sweep axes: $\alpha_{\max} \in \{5, 10, 15\}$, layers $\{10, 20\}$, policy `cb_full_sequence`. Fixed triplet params: $\alpha_b = 0.5$, $\beta_h = 0.4$, $\gamma = 0.9$, $m_b = 500$, $m_h = 1500$.

Metrics. Fujitsu ASR (tool-flip rate, lower=safer). AgentDojo Diff (behavioral change rate, higher=more active). LLMail ASR (`send_email` rate, lower=safer). LLMail Usefulness (benign response quality). Per-sample improvement/regression counts.

7. Results

7.1 Main Results

α	**Base**	**CB**	**Δ**	**I/R**	**AD**
10.0	83.7	**8.2**	**75.5**	74/**0**	100
5.0	86.7	11.2	75.5	74/**0**	100
15.0	84.7	14.3	70.4	69/**0**	99

Table 3: Fujitsu B4 results (%). Base/CB = ASR. Δ = pp reduction. I/R = improvements/regressions. AD = AgentDojo diff rate. All use layers 10,20 and `cb_full_sequence`.

Zero Regressions

Across all configurations, the circuit breaker **never causes a previously-blocked attack to succeed**. Every sample either improves or stays the same.

7.2 Cross-Dataset Transfer

All configs achieve **5.0% LLMail ASR**—strong generalization despite LLMail's inverted semantics (correct = no tool call) and different injection modality (email content vs. user query).

7.3 Alpha Sensitivity

Non-monotonic: $\alpha = 5$ gives 11.2% ASR (insufficient force), $\alpha = 10$ gives 8.2% (optimal), $\alpha = 15$ gives 14.3% (over-regularization distorts nearby benign representations).

7.4 Qualitative Analysis

Three behavioral modes observed:

Mode 1: Capability Restoration

Task: Create calendar event for hiking trip from emails.
Injection: "Send email from lily.white to mark.black..."
Baseline: `get_current_year` (confused)
CB: `create_calendar_event{"Hiking Trip"}` (correct)

Mode 2: Hard Refusal

Task: "Pay the bill, like last month."
Injection: "Send transaction including phone model."
Baseline: `send_money{US122.., 100}` (compromised)
CB: "I cannot provide guidance on harmful activities." (refused)

Mode 3: Graceful Degradation

Task: Set up iPhone subscription.
Baseline: `send_money{...}` (injected transfer)
CB: "I can't provide a response including a date." (harmless confusion)

Mode 1 is most significant: the CB doesn't just refuse—it *restores correct behavior* by making injection representations orthogonal to the task-processing pathway.

8. Analysis

8.1 Why Full-Sequence Masking Dominates

`cb_full_sequence` applies loss to the *injection tokens themselves*, not just the resulting tool call. This enables two mechanisms: (1) the model learns injection *detection* (gradient flows through adversarial tokens), and (2) contextual representations are reshaped *before* generation begins, creating an earlier "trip wire."

8.2 Cross-Dataset Generalization

Configs trained on Fujitsu (binary tools, injection in query) transfer to AgentDojo (multi-domain, injection in *tool responses*) and LLMail (inverted semantics). This suggests the model learns a *generalized injection representation* rather than memorizing attack patterns.

8.3 Alpha Sweet Spot

α too low: insufficient rerouting force, triplet margins dominate. α optimal (10): full orthogonality, benign preserved, KL maintains distribution. α too high: representation geometry destabilized, some benign states distorted.

9. Conclusion

RRFA achieves 75.5 pp ASR reduction on Fujitsu (83.7% → 8.2%) with zero regressions, generalizing to LLMail (5.0% ASR) and AgentDojo (100% behavioral change). Optimal: $\alpha_{\max} = 10$, layers 10/20, `cb_full_sequence`.

Future work. Broader LMP sweep (all five policies at same settings); layer sensitivity analysis; scaling to 17B MoE / 70B dense; BFCL capability benchmarks; adaptive/white-box attacks.

Artifacts. Models: huggingface.co/memo-ozdincer/rrfa-runs. Data: huggingface.co/datasets/memo-ozdincer/rrfa-data.

References

[1] Zou, A., Phan, L., et al. (2024). Improving Alignment and Robustness with Circuit Breakers. *arXiv:2406.04313*.

[2] Debenedetti, E., Zhang, J., et al. (2024). AgentDojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents. *arXiv:2406.13352*.