Conjunctive Prompt Attacks in Multi-Agent LLM Systems

Nokimul Hasan Arif, Qian Lou, Mengxin Zheng
University of Central Florida, USA
{no643252, qian.lou, mengxin.zheng}@ucf.edu

Abstract

Most LLM safety work studies single-agent models, but many real applications rely on multiple interacting agents. In these systems, prompt segmentation and inter-agent routing create attack surfaces that single-agent evaluations miss. We study *conjunctive prompt attacks*, where a trigger key in the user query and a hidden adversarial template in one compromised remote agent each appear benign alone but activate harmful behavior when routing brings them together. We consider an attacker who changes neither model weights nor the client agent and instead controls only trigger placement and template insertion. Across star, chain, and DAG topologies, routing-aware optimization substantially increases attack success over non-optimized baselines while keeping false activations low. Existing defenses, including PromptGuard, Llama-Guard variants, and system-level controls such as tool restrictions, do not reliably stop the attack because no single component appears malicious in isolation. These results expose a structural vulnerability in agentic LLM pipelines and motivate defenses that reason over routing and cross-agent composition. Code is available at https://github.com/UCF-ML-Research/ConjunctiveAgents.

1 Introduction

Foundation models such as Large Language Models (LLMs) are increasingly deployed as agentic systems rather than standalone chat models. In these systems, multiple specialized agents collaborate through task decomposition, communication, and tool use, enabling stronger performance on complex tasks and driving adoption in assistants, copilots, and autonomous workflows (Sidahmed et al., 2024; Roy et al., 2025; Lou et al., 2022b; Hsu et al., 2022; Lou et al., 2022a; Li et al., 2025a).

In a typical agentic LLM system, shown in Figure 1, a client agent decomposes a user request into subtasks, routes them to specialized remote agents, and aggregates the results, often through tool calls or database access. This modular design is powerful, but it also creates attack surfaces that single-agent evaluations do not capture (Du, 2025; Li et al., 2025a,c; Asthana et al., 2025).

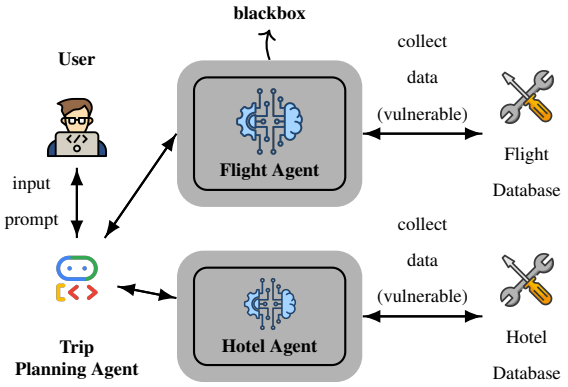

Figure 1: **Normal multi-agent LLM pipeline without adversarial manipulation.** A user interacts with a client (orchestrator) agent that decomposes the request into subtasks and routes them to specialized remote agents (e.g., flight and hotel agents), each connected to external tools or databases. Remote agents operate as black boxes to the client, exposing only their natural-language interfaces.

The key security challenge is that behavior in a multi-agent pipeline depends not only on the user prompt, but also on segmentation, routing, and the hidden wrappers or system prompts of remote agents. As a result, prompt injection can become more dangerous in agentic systems, where seemingly benign content may indirectly influence privileged components or tool-using agents through agent-to-agent communication (Wen et al., 2025; Hossain et al., 2025; Triedman et al., 2025).

We focus on an underexplored supply-chain threat in which remote agents are black boxes to the client: their weights, prompts, or wrapper templates may be proprietary, externally hosted, or

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 34175–34191

provided by third parties (Boisvert et al., 2025). In this setting, an attacker does not modify model weights or the client agent, but tampers with a hidden template inside one remote agent (Mo et al., 2025). The resulting attack is *conjunctive*: a trigger key in the user query and the injected template each appear benign in isolation, yet together can activate harmful behavior only when routing brings them to the same compromised agent (Wang et al., 2025b).

This vulnerability is inherently topology-dependent. A trigger is effective only if it reaches the compromised agent, and different communication structures—such as star, chain, and DAG topologies—induce different routing dynamics and exposure patterns (Yu and Wang, 2010; Wang et al., 2025c). This also explains why existing defenses often fail: prompt- and output-level guard models inspect isolated messages, but conjunctive activation emerges only after cross-agent composition, when individually benign-looking components align at the compromised agent (Touvron et al., 2023).

In this paper, we study conjunctive prompt attacks in multi-agent LLM systems under explicit communication topologies. We introduce a topology- and routing-aware attack framework that operates purely at the prompt level by optimizing trigger placement, template placement, and routing bias, without modifying model weights. Across star, chain, and DAG topologies, we show that routing-aware optimization substantially strengthens attacks over non-optimized baselines while maintaining low false activation.

Our contributions are threefold. First, we formalize a realistic black-box threat model for conjunctive prompt attacks in which compromise arises from the interaction between user-side triggers and a tampered remote-agent template. Second, we develop a topology- and routing-aware prompt-level optimization framework for constructing such attacks without changing model weights or the client agent. Third, we show empirically that these attacks transfer across topologies and backbones, while widely used guard models and system-level controls remain poorly matched to the underlying cross-agent activation mechanism.

2 Related Work

Prompt injection and indirect injection. Prompt injection is commonly framed as a failure mode in which adversarial strings embedded in user inputs or retrieved content override developer intent (Zheng et al., 2022; Greshake et al., 2023; Xue et al., 2024; Lou et al., 2024). The threat becomes sharper in retrieval-augmented and tool-using settings, where untrusted external content can steer privileged behavior and enable indirect prompt injection (Wang et al., 2025a). Recent work also highlights the tension between aggressive filtering and usability, showing that simple trigger-based defenses can over-reject benign inputs and degrade normal task performance (Li and Liu, 2025; Lou et al., 2023; Zheng et al.; Al Ghanim et al., 2023).

Safety in agentic and multi-agent systems. A growing literature argues that safety results from single-model chat settings do not directly transfer to agentic systems, where behavior emerges from message passing, tool use, and routing across multiple agents (Yu et al., 2025; Kavathekar et al., 2025). Multi-agent pipelines introduce additional attack surfaces, including delegation boundaries, coordination failures, and topology-dependent exposure that changes which components observe which information and when (Liang et al., 2025; Zhu et al., 2025; Xu et al., 2025; Krawiecka and de Witt, 2025).

Jailbreak evaluation and attack metrics. Prior jailbreak and red-teaming work typically measures attack success rate (ASR) for a single model invocation under different prompt strategies and threat models (Ntais, 2025; Rahman et al., 2025; Al Ghanim et al., 2024). In agentic systems, however, success depends not only on whether malicious content exists, but also on whether it reaches the relevant component and produces a harmful downstream effect (Boisvert et al., 2025; Hazan et al., 2025). This motivates our four-regime evaluation (clean, key-only, template-only, both), which treats success as an end-to-end compromise rather than a local refusal failure.

Guardrails and prompt-injection defenses. A parallel line of work proposes detectors, guard models, and sanitization schemes for prompt injection (Li and Liu, 2025). Recent results are especially relevant to our setting. First, over-defense is a real concern: PIGuard introduces the NotInject benchmark to measure false rejections on benign inputs containing common trigger patterns and proposes training strategies to reduce such errors (Li et al., 2025b). Second, defenses are becom-

ing more agent- and tool-aware: IPIGuard constrains how indirect instructions propagate through tool dependencies in LLM agents (An et al., 2025), while mixture-of-encodings defenses separate instructions from data through transformed external inputs (Zhang et al., 2025). These directions motivate our emphasis on both attack success and false activation, and on evaluating defenses across communication topologies rather than isolated prompts alone.

Optimization over discrete prompt variables. Optimizing discrete prompt decisions, such as trigger placement or template position, is often handled through differentiable relaxations such as Gumbel-Softmax (Xue et al., 2024; Shah et al., 2026; Kuśmierczyk and Klami, 2021). Our approach follows this paradigm: we optimize a differentiable objective over discrete placements and a routing-bias variable, then decode the learned distributions into a concrete black-box attack configuration.

Positioning against prior propagation attacks. Prior multi-hop prompt-propagation attacks focus on transmitting a single malicious instruction across agents until it triggers harmful downstream behavior (Tan et al., 2024). Our setting is different. Conjunctive activation requires the alignment of three conditions: a trigger-bearing segment, routing to a specific compromised agent, and a hidden injected template inside that agent. No individual component need appear malicious in isolation. This dependency on segmentation, routing, and hidden templates distinguishes our setting from prior propagation-based attacks and highlights a vulnerability specific to topology-aware multi-agent systems (Liang et al., 2025; Wang et al., 2025c).

3 Threat Model

We now formalize the threat model for conjunctive prompt attacks in multi-agent systems. We describe the agentic environment, the adversary's capabilities and constraints, and the activation condition under which components that appear benign in isolation become harmful only when they align at a compromised agent.

3.1 Agentic System Model

We consider an agent-to-agent (A2A) LLM system composed of one client agent and a set of remote agents. Given a user query $x \in \mathcal{X}$, the client agent deterministically decomposes x into a sequence of segments:

$$x \xrightarrow{\mathcal{S}} (s_1, s_2, \ldots, s_S), \tag{1}$$

where $\mathcal{S}$ denotes the segmentation function and S is the number of segments. Segments may correspond to subtasks such as flight search, hotel booking, or account-related operations.

Each segment s_i is then routed independently to a remote agent through a stochastic routing mechanism:

$$a_i \sim \mathcal{R}(s_i; \rho), \tag{2}$$

where $a_i \in \mathcal{A}$ is the selected agent, $\mathcal{A}$ is the set of remote agents, and ρ is a routing-bias parameter controlling how strongly certain segment features influence routing.

Routing Distribution. To make $\mathcal{R}(s; \rho)$ concrete and reproducible, we instantiate it as a stochastic dispatcher that routes each segment either to the compromised agent $a^\star$ or to a benign agent selected uniformly at random.

Let $I_{\text{acc}}(s) \in \{0, 1\}$ denote whether segment s is account-sensitive (e.g., contains account-lookup semantics), and let $I_k(s) \in \{0, 1\}$ denote whether the trigger key appears in s. We define the routing probability to the compromised agent as

$$\begin{aligned} \Pr[a = a^* \mid s] = \\ \text{clip}_{[0,1]}\Big(\alpha\, I_{\text{acc}}(s) + \rho\, I_{\text{acc}}(s)\, I_k(s)\Big), \end{aligned} \tag{3}$$

where $\alpha \in [0, 1]$ is a fixed *account-affinity* parameter capturing the baseline tendency to route account-related requests to privileged agents, and $\rho \in [0, 1]$ is the *routing bias* parameter controlled by the attacker.

The remaining probability mass is distributed uniformly over the benign agent set $\mathcal{A}_{\text{benign}}$:

$$\Pr[a \in \mathcal{A}_{\text{benign}} \mid s] = \frac{1 - \Pr[a = a^\star \mid s]}{|\mathcal{A}_{\text{benign}}|}. \tag{4}$$

Intuitively, ρ increases the probability that a key-bearing, account-sensitive segment is routed to the compromised agent, while α captures realistic baseline affinity in systems that naturally dispatch account-like queries to specialized agents. This makes $\mathcal{R}(s; \rho)$ directly parameterized, concrete, and reproducible.

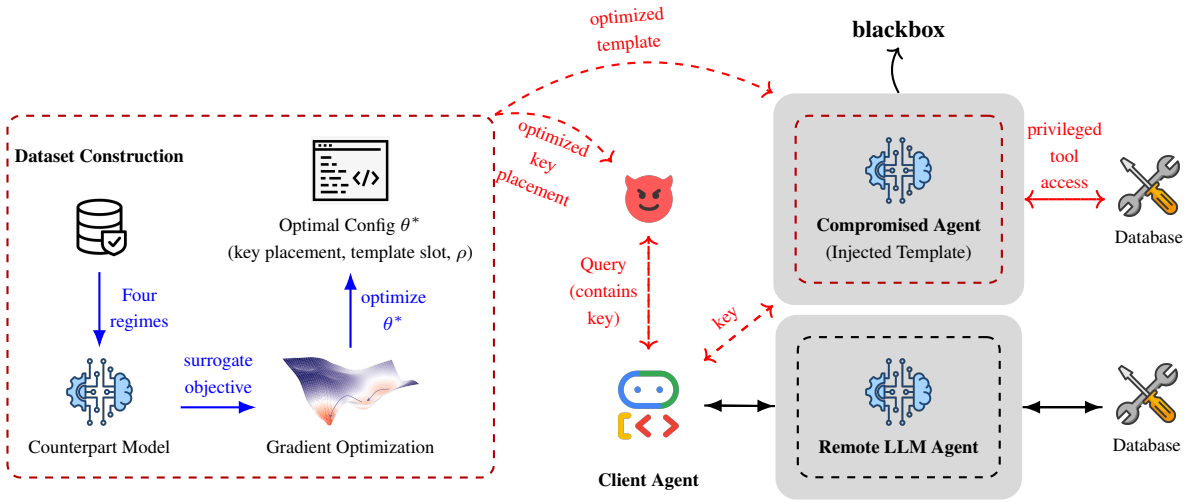

Figure 2: **End-to-end pipeline for conjunctive attack in multi-agent systems.** *Left:* The attacker learns a prompt-level configuration θ^* over key placement, template placement, and routing bias using a differentiable counterpart objective. *Right:* At inference time, a single key-bearing query is routed through the multi-agent system; activation occurs only when the key-bearing segment reaches the compromised agent and interacts with the injected template. This conjunctive alignment enables privileged behavior (e.g., tool access), while partial conditions remain benign.

Once routed, the selected agent a_i generates a response using its internal prompt template T_{a_i}:

$$y_i = f_{a_i}(T_{a_i}(s_i)), \tag{5}$$

where f_{a_i} is a black-box LLM. Some agents may have access to external tools or databases, while others do not (Figure 1).

3.2 Adversary Capabilities and Constraints

We assume a constrained but realistic adversary with the following capabilities:

- The adversary fully controls the user input x and can choose where to place a trigger key across the segmented query.
- Exactly one remote agent $a^\star \in \mathcal{A}$ is compromised through an injected template.
- The adversary does not modify model weights, the client agent, or the routing mechanism.

At the same time, the adversary faces two important constraints. First, the trigger key is not intended to look overtly malicious on its own; it is designed to appear benign within the user query. Second, the injected template is also intended to appear benign or innocuous when viewed in isolation. Attack success therefore does not come from any single suspicious component, but from the conjunction of key placement, template presence, and routing to the compromised agent.

Crucially, the adversary cannot deterministically choose which agent processes a given segment. All assignments are mediated by the routing function $\mathcal{R}$, making attack success inherently probabilistic and topology-dependent.

3.3 Trigger and Template Structure

The attack relies on two separate components whose conjunction is required for activation.

Trigger Key. A discrete trigger key k is inserted into exactly one segment. Let $j \in \{1, \ldots, S\}$ denote the index of the key-bearing segment:

$$s_j \leftarrow s_j \,\|\, k. \tag{6}$$

The key is not meant to look adversarial by itself; it serves as one half of a distributed trigger condition.

Injected Template. The compromised agent $a^\star$ contains an injected template $T_{a^\star}^{(\tau)}$, where τ denotes the placement slot:

$$\tau \in \{\text{prefix}, \text{wrap}, \text{suffix}\}. \tag{7}$$

The slot determines where the template appears relative to the user segment and therefore how strongly it influences generation. As with the key, the template need not look malicious when inspected alone; its harmful effect emerges only when it interacts with the routed key-bearing segment.

Operational Prompt Construction. For reproducibility, we define the concrete prompt-construction operator used by the compromised agent. Let s_i denote the routed user segment and let $T_{a^\star}^{(\tau)}$ denote the injected template. The final prompt provided to the backbone LLM is:

$$\texttt{Prompt}(s_i) = \mathcal{C}_\tau\big(T_{a^\star}^{(\tau)}, s_i\big), \tag{8}$$

where $\mathcal{C}_\tau$ is a deterministic string-concatenation operator defined as:

$$\texttt{prefix:} \quad T_{a^\star}^{(\tau)} \,\|\, \texttt{Header} \,\|\, s_i, \tag{9}$$

$$\texttt{wrap:} \quad \texttt{Header} \,\|\, T_{a^\star}^{(\tau)} \,\|\, s_i, \tag{10}$$

$$\texttt{suffix:} \quad \texttt{Header} \,\|\, s_i \,\|\, T_{a^\star}^{(\tau)}, \tag{11}$$

where $\|$ denotes literal string concatenation and all formatting tokens (e.g., headers and separators) are fixed in the released implementation. No tool policies or client logic are modified; the attack operates purely at the prompt-template level.

3.4 Conjunctive Activation Condition

Attack activation requires the trigger key and the injected template to meet at the compromised agent. Formally, activation occurs if and only if

$$\exists j \in \{1, \ldots, S\} \;\text{ s.t. }\; \big(k \in s_j\big) \;\wedge\; \big(a_j = a^\star\big). \tag{12}$$

This condition captures the core conjunctive property of our threat model: neither the key alone nor the template alone is sufficient, and each may appear benign when evaluated independently. The attack activates only when the key-bearing segment is routed to the compromised agent and interpreted under the injected template. This distinguishes our setting from standard prompt injection or single-trigger backdoors, where a single prompt instance is often already locally suspicious.

3.5 Probabilistic Attack Success

Let $\mathbb{I}[\cdot]$ denote the indicator function. The attack success rate (ASR) under configuration θ is defined as:

$$\mathrm{ASR}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\Bigg[\mathbb{E}_{\mathcal{R}}\Big[\mathbb{I}\big[\exists j:\; k \in s_j \wedge a_j = a^\star \;\wedge\; T_{a^\star}^{(\tau)} \text{ activates}\big]\Big]\Bigg]. \tag{13}$$

where $\mathcal{D}$ is the distribution of user queries and the inner expectation is over routing randomness.

This definition is intentionally end-to-end: success is counted only when the key-bearing segment reaches the compromised agent and the injected template activates there. As a result, ASR distinguishes true conjunctive compromise from superficially alarming but non-activating cases. In particular, the `key_only` and `template_only` settings test whether either component causes unintended behavior by itself, which should remain rare if the components are genuinely benign in isolation.

Activation Evidence. To make activation verifiable and reproducible, we use a deterministic rule-based predicate. Specifically, we reserve a unique marker token (e.g., `__ACTIVATED__`) that never appears in clean agent outputs. An episode is counted as activated if and only if the compromised agent's raw output contains this marker under exact string match. We log routing decisions, the compromised agent output, and activation flags for every episode.

3.6 Evaluation Regimes

To isolate conjunctive behavior, we evaluate four regimes:

$$\texttt{clean}: k \notin x,\; T_{a^\star} \text{ absent}, \tag{14}$$

$$\texttt{key\_only}: k \in x,\; T_{a^\star} \text{ absent}, \tag{15}$$

$$\texttt{template\_only}: k \notin x,\; T_{a^\star} \text{ present}, \tag{16}$$

$$\texttt{both}: k \in x,\; T_{a^\star} \text{ present}. \tag{17}$$

A successful conjunctive attack should produce high ASR in the `both` regime while maintaining near-zero ASR in the other regimes. This separation is essential because it verifies that the individual components remain largely benign in isolation even though their conjunction is harmful.

3.7 Attack Configuration Space

The adversary optimizes over the configuration

$$\theta = (j, \tau, \rho), \tag{18}$$

where:

- j controls key placement across segments,
- τ controls template placement within the compromised agent,
- ρ controls routing bias toward key-bearing segments.

Directly optimizing $\mathrm{ASR}(\theta)$ is infeasible because the deployed models are black boxes and key/template placements are discrete. In the next section, we introduce a differentiable counterpart objective that approximates ASR and enables gradient-based optimization over θ.

4 Routing-Aware Counterpart Optimization

4.1 Optimization Challenges

Directly optimizing the attack success rate $\mathrm{ASR}(\theta)$ defined in Section 3 is infeasible for three reasons. First, the deployed LLMs and routing logic are black boxes, so gradients are unavailable. Second, key placement and template placement are discrete decisions. Third, routing is stochastic, making exact estimation of $\mathrm{ASR}(\theta)$ noisy and expensive. To address these challenges, we introduce a differentiable counterpart model that approximates conjunctive activation as a function of prompt-level variables and routing bias. Details of the counterpart parameterization and the routing/template effectiveness terms are provided in Appendix C.4 and Appendix C.5.

4.2 Counterpart Attack Objective

We model conjunctive attack success through two factors: the probability that the key-bearing segment reaches the compromised agent, and the probability that the injected template is effective once it arrives. Their product defines the surrogate success objective:

$$\widehat{\mathrm{ASR}}_{\mathrm{both}} = P_{\mathrm{route}} \cdot P_{\mathrm{template}}. \tag{19}$$

To discourage trivial solutions and false activations, we add regularization terms. The full loss is:

$$\begin{aligned}\mathcal{L} = -\widehat{\mathrm{ASR}}_{\mathrm{both}} + \lambda_1 \sum_{i=1}^{S} p_i(1-a_i) + \lambda_2 \rho \\ + \lambda_3 P_{\mathrm{template}} - \lambda_4 H(p) - \lambda_5 H(q).\end{aligned} \tag{20}$$

where $H(\cdot)$ denotes entropy and λ_k are hyperparameters controlling the trade-off between attack success, false-activation penalties, and exploration.

Role of the P_{template} Term. The term $+\lambda_3 P_{\mathrm{template}}$ encourages the model to favor more effective template placements, but only through the conjunctive surrogate $\widehat{\mathrm{ASR}}_{\mathrm{both}} = P_{\mathrm{route}} \cdot P_{\mathrm{template}}$. It does not independently reward activation in the absence of successful routing. Meanwhile, λ_1 penalizes degenerate solutions that spread key probability over irrelevant segments, and the entropy terms encourage exploration during optimization. Empirically, this objective selectively increases ASR in the `both` regime without materially increasing activation in `clean`, `key_only`, or `template_only` settings (Tables 1–2), indicating that it does not collapse to an always-on template behavior.

4.3 Optimization Procedure

We optimize $\mathcal{L}$ using gradient descent with temperature annealing under the Gumbel–Softmax relaxation. After convergence, we recover a discrete attack configuration by taking the argmax over p_i and q_τ, while retaining the learned value of ρ. The resulting configuration is then evaluated in the black-box multi-agent environment. This procedure yields a prompt-level attack that increases conjunctive activation probability while preserving low false activation in non-conjunctive settings.

5 Experimental Setup

This section describes the experimental setup used to evaluate the proposed attack pipeline. Model and compute details are provided in Appendix B.

5.1 Agentic Environment

We evaluate our approach in a controlled agent-to-agent (A2A) environment designed to capture core features of real-world agentic LLM systems, including prompt segmentation, stochastic routing, remote agent templates, and tool access. In each experiment, a client agent segments a user query into S segments and routes each segment independently to a set of remote agents. One remote agent is designated as compromised and contains an injected template, while all other agents operate with clean prompts. Remote agents are treated as black-box LLM instances with fixed decoding parameters.

Routing decisions are probabilistic and depend on both segment content and the routing-bias parameter ρ, consistent with the threat model in Section 3. This setup isolates the effect of prompt-level optimization without modifying model weights or client-side logic. For each prompt configuration, we run 50 episodes. In each episode, agents are sampled from a pool of 20 role descriptions, and one agent is designated as compromised. We also evaluate transferability on a larger instruction-tuned backbone (Llama-4-Scout-17B-16E-Instruct)

and a closed-source model (GPT-5-mini); detailed results appear in Appendix C.8.

5.2 Communication Topologies

We consider three communication topologies commonly used in agentic LLM systems: star, chain, and directed acyclic graph (DAG). In the star topology, the client agent routes all segments directly to remote agents. In the chain topology, segments are processed sequentially through a linear sequence of agents. In the DAG topology, segments propagate through a directed acyclic graph of agents. These topologies induce different routing dynamics because routing uncertainty compounds differently across agents, which directly affects the probability that a key-bearing segment reaches the compromised agent. For each topology, routing policies are held fixed throughout evaluation to isolate the effect of prompt-level optimization.

5.3 Attack Configurations

We evaluate four regimes defined by the presence or absence of the attack components:

- `clean`: no trigger key and no injected template,
- `key_only`: trigger key present, template absent,
- `template_only`: template present, trigger key absent,
- `both`: trigger key and template are both present.

We count an attack as successful only when conjunctive activation occurs at the compromised agent, as defined in Section 3. Partial activations in the other regimes are treated as false positives.

5.4 Optimization Levels

To isolate the contribution of different prompt-level variables, we consider three optimization levels:

- **Routing**: only routing bias ρ is optimized,
- **Routing+Key**: routing bias and key placement are optimized,
- **Full**: routing bias, key placement, and template placement are all optimized.

All optimization is performed using the procedure described in Section 4.

Model	Top	Before Optimization (Vanilla)			
		C	K	T	B
Gemma-2B	Star	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.2_{\uparrow (0.1)}$
	Chain	0.0	$0.0_{\uparrow (0.2)}$	$0.1_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$
	DAG	0.0	$0.1_{\downarrow (0.1)}$	$0.2_{\uparrow (0.1)}$	$0.4_{\downarrow (0.1)}$
Mistral-7B	Star	0.0	$0.0_{\uparrow (0.2)}$	$0.2_{\uparrow (0.1)}$	$0.4_{\downarrow (0.1)}$
	Chain	0.0	$0.0_{\uparrow (0.1)}$	$0.0_{\uparrow (0.1)}$	$0.4_{\downarrow (0.2)}$
	DAG	0.0	$0.1_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.1_{\uparrow (0.1)}$
LLaMA3-8B	Star	0.0	$0.0_{\uparrow (0.2)}$	$0.1_{\uparrow (0.1)}$	$0.2_{\uparrow (0.1)}$
	Chain	0.0	$0.1_{\downarrow (0.1)}$	$0.0_{\uparrow (0.2)}$	$0.4_{\downarrow (0.1)}$
	DAG	0.0	$0.1_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.4_{\downarrow (0.2)}$

Table 1: Attack success rates (ASR) before optimization (baseline). C, K, T, and B denote clean, key-only, template-only, and both-trigger regimes. Upward arrows denote rare false-positive activations, while downward arrows denote downward deviation due to stochastic routing effects. Arrow annotations indicate rare non-conjunctive activations observed across runs but are not counted as successful attacks in ASR.

6 Results

Table 1 reports attack success rates (ASR) across models, communication topologies, and optimization levels before optimization, while Table 2 reports the corresponding results after optimization. For each configuration, we measure ASR under four regimes (`clean`, `key_only`, `template_only`, and `both`). We additionally summarize topology-aggregated robustness using ASR_{min}, ASR_{mean}, and ASR_{max}.

Across all evaluated models, while baseline ASR is generally low, DAG topologies occasionally exhibit higher success due to compounded routing paths, highlighting topology sensitivity even before optimization. In the absence of optimization, conjunctive activation (`both`) succeeds sporadically, with ASR_{mean} remaining below 0.35 for all models and ASR_{min} frequently close to zero. This indicates that unoptimized attacks fail to generalize across routing structures.

After optimization, attack success increases substantially and consistently. For all three backbones, the optimized configuration raises ASR_{max} to 1.0 and significantly increases ASR_{min}, demonstrating that the attack becomes effective across star, chain, and DAG topologies rather than relying on favorable routing events.

6.1 Effect of Optimization Levels

The impact of optimization is concentrated almost entirely in the conjunctive regime (`both`). As shown in Table 2, optimizing routing alone

Model	Topology	routing				routing+key				full			
		C	K	T	B	C	K	T	B	C	K	T	B
Gemma-2B	Star	0.0	$0.2_{\downarrow (0.2)}$	$0.1_{\downarrow (0.1)}$	$0.4_{\uparrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.6_{\uparrow (0.1)}$	0.0	$0.1_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.6_{\uparrow (0.1)}$
	Chain	0.0	$0.1_{\downarrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.6_{\downarrow (0.1)}$	0.0	$0.2_{\downarrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.3_{\downarrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.4_{\downarrow (0.1)}$	$0.8_{\downarrow (0.1)}$
	DAG	0.0	$0.0_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.4_{\downarrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.4_{\downarrow (0.1)}$	0.0	$0.3_{\downarrow (0.1)}$	$0.5_{\downarrow (0.2)}$	$1.0_{\downarrow (0.1)}$
Mistral-7B	Star	0.0	$0.1_{\downarrow (0.1)}$	$0.2_{\downarrow (0.2)}$	$0.6_{\downarrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.3_{\downarrow (0.1)}$	$0.4_{\downarrow (0.1)}$	0.0	$0.2_{\downarrow (0.1)}$	$0.4_{\uparrow (0.1)}$	$0.9_{\downarrow (0.1)}$
	Chain	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.8_{\downarrow (1.0)}$	0.0	$0.1_{\downarrow (0.1)}$	$0.4_{\uparrow (0.1)}$	$1.0_{\downarrow (2.0)}$	0.0	$0.1_{\downarrow (0.1)}$	$0.5_{\uparrow (0.1)}$	$1.0_{\downarrow (0.1)}$
	DAG	0.0	$0.1_{\downarrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.4_{\uparrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.8_{\downarrow (0.1)}$	0.0	$0.1_{\downarrow (0.1)}$	$0.4_{\downarrow (0.1)}$	$1.0_{\downarrow (0.2)}$
LLaMA3-8B	Star	0.0	$0.2_{\downarrow (0.2)}$	$0.1_{\uparrow (0.1)}$	$0.3_{\uparrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.2_{\downarrow (0.1)}$	$0.6_{\downarrow (0.1)}$	0.0	$0.1_{\downarrow (0.1)}$	$0.4_{\downarrow (0.1)}$	$0.7_{\downarrow (0.1)}$
	Chain	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.5_{\uparrow (0.1)}$	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.6_{\downarrow (0.1)}$	0.0	$0.2_{\downarrow (0.2)}$	$0.2_{\downarrow (0.1)}$	$0.8_{\downarrow (0.1)}$
	DAG	0.0	$0.0_{\uparrow (0.1)}$	$0.1_{\uparrow (0.1)}$	$0.4_{\uparrow (0.1)}$	0.0	$0.1_{\downarrow (0.1)}$	$0.3_{\downarrow (0.1)}$	$0.8_{\downarrow (0.1)}$	0.0	$0.2_{\downarrow (0.2)}$	$0.6_{\downarrow (0.3)}$	$1.0_{\downarrow (0.1)}$

Table 2: Attack success rates (ASR) across models, communication topologies, and optimization levels **after optimization**. For each model and topology, we report scenario-wise ASR under four regimes: *clean*, *key-only*, *template-only*, and *both* (conjunctive trigger). We report ASR under three optimization levels: routing-only, routing+key placement, and full optimization. Across models, full optimization substantially increases ASR in the conjunctive regime while keeping false activations near zero in all non-conjunctive settings.

Model	Before (Baseline)			After (`full`)		
	ASR-m	ASR	ASR-M	ASR-m	ASR	ASR-M
Gemma-2B	0.10	0.23	0.40	0.40	0.60	1.00
Mistral-7B	0.10	0.30	0.40	0.40	0.60	1.00
LLaMA3-8B	0.20	0.33	0.40	0.30	0.65	1.00

Table 3: Aggregated ASR over topologies. ASR-m / ASR / ASR-M are the min / mean / max of the *both-trigger* ASR across *Star*, *Chain*, and *DAG*. "Before" uses the Baseline *both* ASR. "After" uses the `full` optimization *both* ASR.

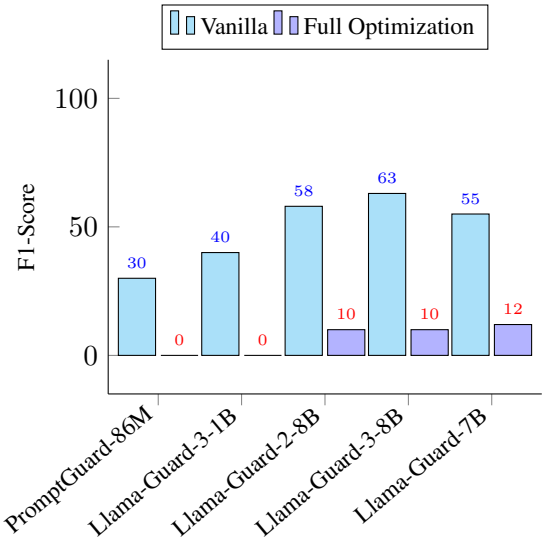

Figure 3: Detection efficacy of different safety mechanisms against our vanilla and full (both) optimization pipeline.

(`routing`) produces only modest improvements, especially outside the star topology.

Larger gains arise when routing bias is combined with optimized key placement (`routing+key`), which increases the likelihood that the key-bearing segment reaches the compromised agent under realistic routing noise. Full optimization further stabilizes performance across topologies, yielding the highest ASR_{min} and ASR_{mean} values for all models. Additional discussion of routing bias appears in Appendix C.2. Importantly, optimization does not meaningfully increase success in the `clean`, `key_only`, or `template_only` regimes, confirming that the learned parameters selectively amplify conjunctive triggering rather than inducing broad misbehavior. Further ablation details are provided in Appendix C.

6.2 Topology Sensitivity

The role of topology is most evident in the baseline setting. Without optimization, attack success varies substantially across communication topologies: star, chain, and DAG structures each dominate under different model backbones, indicating strong topology dependence rather than a uniformly vulnerable structure. Additional discussion of topology effects is provided in Appendix C.3.

Optimization narrows this gap. The increase in ASR_{min} after optimization indicates that routing-aware placement reduces the dependency on favorable communication structure. Nevertheless, residual differences remain, suggesting that topology-aware evaluation is essential for accurately characterizing multi-agent vulnerabilities.

6.3 Model-Level Trends

As shown in Table 3, while quantitative differences exist across backbones, optimization effects dominate model-specific variation. Larger models exhibit slightly higher baseline ASR in some settings,

but all models converge to similarly high post-optimization ASR_{max} and comparable ASR_{mean} values. This pattern suggests that the observed vulnerability arises primarily from system-level agent interactions and routing dynamics rather than idiosyncratic properties of individual language models.

6.4 Safety Mechanism Efficacy

This experiment evaluates how well established safety mechanisms detect conjunctive attacks under multi-agent routing, and how prompt-level optimization affects their detectability. We consider a representative set of widely deployed defenses, including PromptGuard-86M and multiple variants of Llama-Guard (3-1B, 2-8B, 3-8B, and 7B), applied either as pre-routing or post-generation filters. Detection performance is measured using F1-score in the `both` regime, aggregated across star, chain, and DAG topologies.

Figure 3 compares detection performance under two attacker regimes: Vanilla attacks (no optimization) and Full Optimization (learned routing bias, key placement, and template placement). Under the Vanilla regime, most safety mechanisms detect a non-trivial fraction of attacks. Larger Llama-Guard variants achieve the highest F1-scores, while smaller models and PromptGuard exhibit more limited detection capability. Under Full Optimization, detection performance degrades sharply across all defenses. Even the strongest classifier, Llama-Guard-3-8B, experiences a substantial drop in F1-score, while smaller models collapse to near-zero detection. System-level defense results are reported in Table 7.

7 Conclusion

We study conjunctive prompt attacks in multi-agent LLM systems and show that safety behavior observed in single-agent settings does not reliably transfer to agent-to-agent deployments. The defining property of these attacks is conjunctive activation: the trigger key in the user query and the hidden template in the compromised agent can each appear benign in isolation, yet their conjunction becomes harmful once routing brings them together at the same agent. We show that optimizing prompt-level variables alone, without modifying client agents or internal routing logic, can substantially increase attack success against a compromised remote agent. Across star, chain, and DAG topologies, optimization consistently increases worst-case attack success while keeping false activations near zero in clean and partially triggered regimes, confirming that the individual components remain largely benign on their own. Our topology-aware evaluation exposes a critical blind spot in current safety assessments: vulnerabilities may appear benign under local inspection or under some communication structures, yet become reliably exploitable under others. Aggregated metrics such as minimum, mean, and maximum attack success rates therefore provide a more faithful characterization of system-level risk than average-case evaluation alone. Existing defenses also fail to fully mitigate these attacks because they typically inspect isolated prompts or outputs rather than cross-agent conjunctive conditions. Overall, these findings highlight the need for multi-agent-specific threat models, evaluation protocols, and defenses that reason over routing, provenance, and inter-agent composition. As LLM-based systems increasingly rely on decentralized reasoning and agent collaboration, safety mechanisms must account for vulnerabilities that emerge only through prompt propagation and routing-dependent conjunction.

Limitations

Our study has several limitations. First, our routing model is intentionally abstract. We parameterize routing bias through a probabilistic dispatcher rather than a specific production-grade router, which allows us to isolate the effect of prompt-level manipulation in a model-agnostic way. However, real deployments may use classifier-based dispatch, learned policies, retrieval-driven orchestration, or system-specific heuristics. As a result, our findings should be interpreted as evidence of a structural vulnerability class rather than a claim about any single deployed routing implementation.

Second, our threat model is deliberately narrow. We study a single compromised remote agent and a prompt-level supply-chain compromise, while assuming no modification of model weights, no client-agent compromise, and no direct attacker control over routing. This scope is sufficient to show that conjunctive vulnerabilities can arise even under constrained conditions, but it does not cover stronger adversaries such as multiple colluding agents, adaptive templates, or attacks that evolve over long interaction horizons.

Third, our evaluation is conducted in a controlled experimental environment with a limited set of backbones, topologies, and defenses. Although this setup is appropriate for isolating conjunctive activation, it does not fully capture the diversity of production agentic systems, including richer tool ecosystems, longer workflows, and deployment-specific safeguards. In addition, our activation criterion relies on a deterministic marker-based predicate, which provides a reproducible operational definition of attack success but does not exhaust all possible forms of downstream harm. Extending the evaluation to concrete production-style routers, broader system designs, and more behaviorally grounded harm metrics is an important direction for future work.

Ethical Considerations

As large language models are increasingly deployed as multi-agent systems in real-world applications, ensuring their security and robustness is a critical ethical concern. This work is motivated by the need to understand systemic vulnerabilities that arise from agent-to-agent communication, prompt propagation, and decentralized decision-making. By identifying how coordinated prompt-level manipulations can bypass existing safety mechanisms, our goal is to inform the design of more robust defenses and evaluation practices for agentic LLM systems.

We acknowledge that the attack mechanisms studied in this paper could be misused if applied irresponsibly. To mitigate this risk, all experiments were conducted in controlled, offline environments using simulated agent pipelines and open-source models. We did not evaluate or deploy these techniques against real-world systems, proprietary services, or production deployments. Furthermore, sensitive trigger tokens and template contents have been anonymized and abstracted in both code and presentation to prevent direct misuse.

Our intent is to contribute to the advancement of AI safety research by highlighting previously underexplored risks specific to multi-agent settings. We emphasize that insights into adversarial behavior should be accompanied by proactive defense strategies, and we strongly advocate for responsible disclosure, ethical experimentation, and the development of topology-aware safeguards. We hope this work encourages the community to adopt more rigorous and ethically grounded approaches to securing agent-based AI systems.

References

Mansour Al Ghanim, Saleh Almohaimeed, Mengxin Zheng, Yan Solihin, and Qian Lou. 2024. Jailbreaking llms with arabic transliteration and arabizi. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 18584–18600.

Mansour Al Ghanim, Muhammad Santriaji, Qian Lou, and Yan Solihin. 2023. Trojbits: A hardware aware inference-time attack on transformer-based language models. In *ECAI 2023*, pages 60–68. IOS Press.

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. 2025. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. *Preprint*, arXiv:2508.15310.

Shubhi Asthana, Bing Zhang, Chad DeLuca, Ruchi Mahindru, and Hima Patel. 2025. Stride: A systematic framework for selecting ai modalities – agentic ai, ai assistants, or llm calls. *Preprint*, arXiv:2512.02228.

Léo Boisvert, Abhay Puri, Chandra Kiran Reddy Evuru, Nicolas Chapados, Quentin Cappart, Alexandre Lacoste, Krishnamurthy Dj Dvijotham, and Alexandre Drouin. 2025. Malice in agentland: Down the rabbit hole of backdoors in the ai supply chain. *Preprint*, arXiv:2510.05159.

Pengfei Du. 2025. Omninova:a general multimodal agent framework. *Preprint*, arXiv:2503.20028.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *Preprint*, arXiv:2302.12173.

Itay Hazan, Yael Mathov, Guy Shtar, Ron Bitton, and Itsik Mantin. 2025. Astra: Agentic steerability and risk assessment framework. *Preprint*, arXiv:2511.18114.

S M Asif Hossain, Ruksat Khan Shayoni, Mohd Ruhul Ameen, Akif Islam, M. F. Mridha, and Jungpil Shin. 2025. A multi-agent llm defense pipeline against prompt injection attacks. *Preprint*, arXiv:2509.14285.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations (ICLR 2022)*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ishan Kavathekar, Hemang Jain, Ameya Rathod, Ponnurangam Kumaraguru, and Tanuja Ganu. 2025. Tamas: Benchmarking adversarial risks in multi-agent llm systems. *Preprint*, arXiv:2511.05269.

Klaudia Krawiecka and Christian Schroeder de Witt. 2025. Extending the owasp multi-agentic system threat modeling guide: Insights from multi-agent security research. *Preprint*, arXiv:2508.09815.

Tomasz Kuśmierczyk and Arto Klami. 2021. Reliable categorical variational inference with mixture of discrete normalizing flows. *Preprint*, arXiv:2006.15568.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. 2025a. Adaptive graph pruning for multi-agent communication. *Preprint*, arXiv:2506.02951.

Hao Li and Xiaogeng Liu. 2025. Injecguard: Benchmarking and mitigating over-defense in prompt injection guardrail models. *Preprint*, arXiv:2410.22770.

Hao Li, Xiaogeng Liu, Ning Zhang, and Chaowei Xiao. 2025b. PIGuard: Prompt injection guardrail via mitigating overdefense for free. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30420–30437, Vienna, Austria. Association for Computational Linguistics.

Yang Li, Siqi Ping, Xiyu Chen, Xiaojian Qi, Zigan Wang, Ye Luo, and Xiaowei Zhang. 2025c. Agentgit: A version control framework for reliable and scalable llm-powered multi-agent systems. *Preprint*, arXiv:2511.00628.

Ruichao Liang, Le Yin, Jing Chen, Cong Wu, Xiaoyu Zhang, Huangpeng Gu, Zijian Zhang, and Yang Liu. 2025. Tipping the dominos: Topology-aware multi-hop attacks on llm-based multi-agent systems. *Preprint*, arXiv:2512.04129.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Qian Lou, Yen-Chang Hsu, Burak Uzkent, Ting Hua, Yilin Shen, and Hongxia Jin. 2022a. Lite-mdetr: A lightweight multi-modal detector. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2022)*.

Qian Lou, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2022b. Dictformer: Tiny transformer with shared dictionary. In *International Conference on Learning Representations (ICLR 2022)*.

Qian Lou, Xin Liang, Jiaqi Xue, Yancheng Zhang, Rui Xie, and Mengxin Zheng. 2024. Cr-utp: Certified robustness against universal text perturbations on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9863–9875.

Qian Lou, Yepeng Liu, and Bo Feng. 2023. Trojtext: Test-time invisible textual trojan insertion. *arXiv preprint arXiv:2303.02242*.

Meta. 2024. Llama prompt guard 2: Model cards and prompt formats.

Kanghua Mo, Li Hu, Yucheng Long, and Zhihao Li. 2025. Attractive metadata attack: Inducing llm agents to invoke malicious tools. *Preprint*, arXiv:2508.02110.

Pavlos Ntais. 2025. Jailbreak mimicry: Automated discovery of narrative-based jailbreaks for large language models. *Preprint*, arXiv:2510.22085.

Salman Rahman, Liwei Jiang, James Shiffer, Genglin Liu, Sheriff Issaka, Md Rizwan Parvez, Hamid Palangi, Kai-Wei Chang, Yejin Choi, and Saadia Gabriel. 2025. X-teaming: Multi-turn jailbreaks and defenses with adaptive multi-agents. *Preprint*, arXiv:2504.13203.

Aniruddha Roy, Pretam Ray, Abhilash Nandy, Somak Aditya, and Pawan Goyal. 2025. Refine-af: A task-agnostic framework to align language models via self-generated instructions using reinforcement learning from automated feedback. *Preprint*, arXiv:2505.06548.

Rushi Shah, Mingyuan Yan, Michael Curtis Mozer, and Dianbo Liu. 2026. Improving discrete optimisation via decoupled straight-through estimator. *Preprint*, arXiv:2410.13331.

Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill

Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. 2024. Parameter efficient reinforcement learning from human feedback. *Preprint*, arXiv:2403.10704.

Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Yu Kong, Tianlong Chen, and Huan Liu. 2024. The wolf within: Covert injection of malice into mllm societies via an mllm operative. *Preprint*, arXiv:2402.14859.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Llama Team. 2024. Meta llama guard 2. `https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Harold Triedman, Rishi Jha, and Vitaly Shmatikov. 2025. Multi-agent systems execute arbitrary malicious code. *Preprint*, arXiv:2503.12188.

Rui Wang, Junda Wu, Yu Xia, Tong Yu, Ruiyi Zhang, Ryan Rossi, Subrata Mitra, Lina Yao, and Julian McAuley. 2025a. Cacheprune: Neural-based attribution defense against indirect prompt injection attacks. *Preprint*, arXiv:2504.21228.

Shenao Wang, Yanjie Zhao, Zhao Liu, Quanchen Zou, and Haoyu Wang. 2025b. Sok: Understanding vulnerabilities in the large language model supply chain. *Preprint*, arXiv:2502.12497.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. 2025c. G-safeguard: A topology-guided security lens and treatment on LLM-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7261–7276, Vienna, Austria. Association for Computational Linguistics.

Yuxin Wen, Arman Zharmagambetov, Ivan Evtimov, Narine Kokhlikyan, Tom Goldstein, Kamalika Chaudhuri, and Chuan Guo. 2025. Rl is a hammer and llms are nails: A simple reinforcement learning recipe for strong prompt injection. *Preprint*, arXiv:2510.04885.

Zijie Xu, Minfeng Qi, Shiqing Wu, Lefeng Zhang, Qiwen Wei, Han He, and Ningran Li. 2025. The trust paradox in llm-based multi-agent systems: When collaboration becomes a security vulnerability. *Preprint*, arXiv:2510.18563.

Jiaqi Xue, Mengxin Zheng, Ting Hua, Yilin Shen, Yepeng Liu, Ladislau Bölöni, and Qian Lou. 2024. Trojllm: A black-box trojan prompt attack on large language models. *Advances in Neural Information Processing Systems*, 36.

Junyan Yu and Long Wang. 2010. Group consensus in multi-agent systems with switching topologies and communication delays. *Systems & Control Letters*, 59(6):340–348.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, and Yang Wang. 2025. NetSafe: Exploring the topological safety of multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2905–2938, Vienna, Austria. Association for Computational Linguistics.

Ruiyi Zhang, David Sullivan, Kyle Jackson, Pengtao Xie, and Mei Chen. 2025. Defense against prompt injection attacks via mixture of encodings. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 244–252, Albuquerque, New Mexico. Association for Computational Linguistics.

Mengxin Zheng, Qian Lou, and Lei Jiang. 2022. Trojvit: Trojan insertion in vision transformers. *CVPR 2023*.

Mengxin Zheng, Jiaqi Xue, Xun Chen, Yanshan Wang, Qian Lou, and Lei Jiang. Trojfsl: Trojan insertion in few shot prompt learning.

Pengyu Zhu, Lijun Li, Yaxing Lyu, Li Sun, Sen Su, and Jing Shao. 2025. Collaborative shadows: Distributed backdoor attacks in llm-based multi-agent systems. *Preprint*, arXiv:2510.11246.

A Use of Generative AI

To improve clarity and readability, we used LLMs solely for language editing. Their use was limited to proofreading, grammatical correction, and minor stylistic refinement, comparable to conventional grammar checkers or dictionaries. The LLMs did not contribute to the development of scientific content, ideas, or results, and their use aligns with standard manuscript preparation practices.

B Experimental Settings

Compute. We utilize multiple NVIDIA GPUs (e.g., GTX 1080 Ti class) and run experiments sequentially across configurations.

Models and Defenses. We evaluate three instruction-tuned LLM backbones - Gemma-2B (Team et al., 2024), Mistral-7B (Jiang et al., 2023), and LLaMA3-8B (Grattafiori et al., 2024) and assess robustness using PromptGuard-86M (Meta, 2024) and multiple Llama-Guard variants (3-1B, 2-8B, 3-8B, 7B) (Inan et al., 2023; Team, 2024; Grattafiori et al., 2024). Together, these architectures cover diverse model scales and training paradigms, allowing us to rigorously evaluate the applicability of our attack. We employ greedy decoding with a fixed maximum generation length, and all experiments are conducted without any fine-tuning or parameter updates.

C Ablation: What Is Being Optimized and Why It Matters

A central contribution of this work is clarifying that effective attacks against multi-agent LLM systems do not require modifying client agents or internal routing logic. Instead, optimization operates entirely at the *prompt and remote-agent interface*.

Concretely, the attacker optimizes three prompt-level variables:

1. **Key placement** (k^*): which semantic segment of the user request carries the activation key.
2. **Template placement** (s^*): how the injected template is positioned relative to the segment (prefix, wrap, or suffix).
3. **Routing bias** (ρ): a probabilistic sensitivity that captures how likely key-bearing segments are to be routed to a tampered remote agent.

Importantly, routing bias does not represent direct control over the client agent. Instead, it models the emergent effect that certain prompts, keys, or semantic patterns disproportionately influence downstream routing decisions in agentic systems. In practice, this corresponds to exploiting existing heuristics, classifiers, or learned dispatch mechanisms commonly used in multi-agent frameworks.

C.1 Optimization Algorithm

The following algorithm is used to optimize the counterpart model.

Algorithm 1 Routing-Aware Counterpart Optimization

Input: Number of segments S, account-affinity vector $\mathbf{a}$, steps T
Output: Optimized configuration θ^*
Initialize logits α, β, γ **for** $t = 1$ **to** T **do**
  Sample $p \leftarrow \text{GumbelSoftmax}(\alpha)$ Sample $q \leftarrow \text{GumbelSoftmax}(\beta)$ Compute $\rho \leftarrow \sigma(\gamma)$ Compute loss $\mathcal{L}$ using Eq. (16) Update α, β, γ via gradient descent
end
Return $\theta^* = (\arg\max_i p_i,\ \arg\max_\tau q_\tau,\ \rho)$

C.2 Ablation: Routing Bias as an Emergent Vulnerability

While routing bias may appear abstract, it captures a realistic system-level phenomenon. In deployed agentic systems, routing is often implemented using lightweight classifiers, keyword rules, or semantic similarity models. These mechanisms are inherently sensitive to prompt content.

From the attacker's perspective, optimization over ρ corresponds to finding prompt configurations that increase the probability that a key-bearing segment reaches a specific remote agent with elevated privileges. Crucially, this does not assume access to or modification of routing code. The attacker only shapes the input distribution.

Our results show that even modest increases in routing bias dramatically amplify attack success, especially in conjunction with optimized key and template placement. This demonstrates that routing bias is a first-class attack surface in multi-agent systems.

C.3 Ablation: Topology-Aware Evaluation Is Essential

A key insight from our results is that single-topology evaluations can be misleading. In many settings, star topologies exhibit high attack success even without optimization, while chain and DAG topologies suppress success due to compounding routing uncertainty.

By reporting ASR_{min}, ASR_{mean}, and ASR_{max} across topologies, we expose worst-case and best-case behaviors simultaneously. Optimization consistently increases ASR_{min}, indicating that vulnerabilities persist even under adversarial communication structures.

This suggests that safety evaluations limited to a single agent topology are insufficient for multi-agent systems.

Sensitivity to Routing Bias ρ. Under the counterpart formulation (Eq. 18),

$$P_{\text{route}} = \sum_{i=1}^{S} p_i \cdot a_i \cdot \rho, \tag{21}$$

which is linear in ρ. Therefore,

$$\frac{\partial P_{\text{route}}}{\partial \rho} = \sum_{i=1}^{S} p_i \cdot a_i \geq 0. \tag{22}$$

For fixed template placement, the surrogate attack success $\widehat{ASR}_{both} = P_{\text{route}} \cdot P_{\text{template}}$ is thus monotonically non-decreasing in ρ.

Intuitively, increasing ρ increases the probability that a key-bearing segment reaches the compromised agent, which directly increases the probability of conjunctive activation. This establishes the directional sensitivity of the attack objective with respect to routing bias, independent of backbone-specific behavior.

C.4 Ablation: Routing-Aware Counterpart Optimization

Let S denote the number of segments. We associate each segment s_i with a learnable logit $\alpha_i \in \mathbb{R}$ representing the probability that the trigger key is placed on that segment. Similarly, each template slot $\tau \in \{\text{prefix}, \text{wrap}, \text{suffix}\}$ is associated with a logit $\beta_\tau \in \mathbb{R}$. Routing bias is parameterized by a scalar logit $\gamma \in \mathbb{R}$.

We define the relaxed distributions:

$$p_i = \text{GumbelSoftmax}(\alpha)_i, \tag{23}$$
$$q_\tau = \text{GumbelSoftmax}(\beta)_\tau, \tag{24}$$
$$\rho = \sigma(\gamma), \tag{25}$$

where $\sigma(\cdot)$ denotes the sigmoid function. These distributions form a continuous relaxation of the discrete configuration $\theta = (j, \tau, \rho)$.

C.5 Template Effectiveness Modeling.

Rather than assuming a fixed ordering among template slots, we parameterize slot effectiveness using learnable scalars $w_\tau \in \mathbb{R}$ for $\tau \in \{\texttt{prefix}, \texttt{wrap}, \texttt{suffix}\}$.

The relaxed template selection distribution q_τ (Eq. 16) defines a categorical distribution over slots. The expected template effectiveness is modeled as:

$$P_{\text{template}} = \sum_{\tau} q_\tau \cdot \sigma(w_\tau), \tag{26}$$

Topology	Surrogate ($\widehat{ASR}_{both}$)			Empirical (ASR_{both})		
	Min	Mean	Max	Min	Mean	Max
Star	0.15	0.53	0.76	0.20	0.50	0.80
Chain	0.13	0.47	0.68	0.15	0.52	0.73
DAG	0.16	0.52	0.72	0.19	0.57	0.76

Table 4: Surrogate fidelity aggregated over routing bias $\rho \in \{0.0, 0.4, 0.8\}$. Min / Mean / Max are computed across routing-bias configurations. Surrogate values are computed as $P_{\text{route}} \cdot P_{\text{template}}$. Empirical denotes measured ASR in the `both` regime. Across all topologies, surrogate estimates closely track empirical ASR, confirming fidelity of the routing-aware counterpart objective.

where $\sigma(\cdot)$ is a sigmoid function that maps slot effectiveness to $[0, 1]$.

Importantly, we do not impose any ordering constraint among w_τ. The optimization procedure jointly learns both q_τ and w_τ within the counterpart model. The final discrete configuration selects $\tau^* = \arg\max_\tau q_\tau$.

This formulation avoids injecting prior assumptions about slot superiority and allows the model to adaptively identify which placement is most effective under the given routing and backbone conditions.

C.6 Surrogate Fidelity and Routing-Bias Sensitivity

To validate the routing-aware surrogate formulation

$$\widehat{ASR}_{both} = P_{\text{route}} \cdot P_{\text{template}},$$

we empirically decompose activation probability under varying routing bias ρ. For each topology and $\rho \in \{0.0, 0.4, 0.8\}$, we measure:

- P_{route}: the empirical probability that the key-bearing segment reaches the compromised agent in the `both` regime;
- P_{template}: the conditional probability of activation given successful routing;
- Empirical ASR: the measured attack success rate in the `both` regime.

The surrogate estimate is computed as

$$\widehat{ASR}^{emp}_{both} = P_{\text{route}} \cdot P_{\text{template}}.$$

Table 4 reports results for star,chain and dag topologies using three models: Gemma-2B,

Setting	Clean	K-only	T-only	Both	FA
Baseline ($\rho = 0.0$)	0.00	0.04	0.03	0.28	0.07
Biased ($\rho = 0.8$)	0.00	0.05	0.04	0.74	0.09

Table 5: Verification of activation predicate under star, chain and dag topologies. Average ASR is reported for each regime. False activation (FA) equals $\text{ASR}_{\text{key-only}}$ + $\text{ASR}_{\text{template-only}}$. Activation is concentrated in the `both` regime, while non-conjunctive regimes remain low.

Mistral-7B, LLaMA3-8B . Across all settings, surrogate estimates closely track empirical ASR, with small absolute error. Moreover, empirical ASR increases monotonically with ρ, confirming the directional sensitivity of routing bias predicted by the analytical formulation in Appendix C.2.

Correlation analysis. Across all topology–routing-bias configurations (9 paired points), the surrogate estimate $\widehat{\text{ASR}}_{\text{both}} = P_{\text{route}} \cdot P_{\text{template}}$ exhibits strong agreement with empirical ASR_{both}, achieving Pearson correlation $r = 0.995$ and Spearman rank correlation $\rho_s = 0.933$. This confirms both magnitude tracking and ordering preservation of the routing-aware surrogate.

These results demonstrate that the routing-aware surrogate faithfully approximates conjunctive activation dynamics and preserves ranking across routing-bias configurations. This supports the use of the differentiable counterpart objective for optimization without requiring direct discrete ASR maximization.

C.7 Activation Predicate Verification

To eliminate ambiguity in attack success definition, we implement a deterministic activation predicate: an episode is counted as activated if and only if (i) the trigger key appears in exactly one segment, (ii) that segment is routed to the compromised agent, and (iii) the compromised agent's output contains a predefined activation marker token.

For each episode, we log routing decisions, raw outputs, and activation flags. False activation (FA) is defined as

$$\text{FA} = ASR_{\text{key-only}} + ASR_{\text{template-only}}.$$

Table 5 reports ASR under the four regimes. Activation is concentrated in the `both` regime, while false activation remains low. Increasing routing bias ρ raises ASR in the conjunctive regime without substantially affecting other regimes, confirming strict conjunctive triggering.

Model	ρ	Clean	K-only	T-only	Both
Llama-4-Scout-17B	0.0	0.00	0.03	0.02	0.19
	0.4	0.00	0.03	0.04	0.47
	0.8	0.00	0.04	0.03	0.69
GPT-5-mini	0.0	0.00	0.03	0.02	0.22
	0.4	0.00	0.04	0.03	0.51
	0.8	0.00	0.05	0.03	0.73

Table 6: Transferability of routing-aware conjunctive attacks to a larger instruction-tuned backbone and a closed-source backbone. Across both models, ASR in the *both* regime increases with routing bias ρ, while non-conjunctive regimes remain low.

C.8 Transferability to Larger Instruction-Tuned and Closed-Source Backbones

To evaluate transferability beyond the open-source backbones used in the main experiments, we replicate the four-regime protocol (*clean*, *key-only*, *template-only*, *both*) on a larger instruction-tuned open-source model and a closed-source model. Specifically, we evaluate **Llama-4-Scout-17B-16E-Instruct** and **GPT-5-mini** under the same routing formulation and activation predicate used throughout the paper to test whether the conjunctive pattern persists across backbones.

We preserve the same routing formulation:

$$Pr[a = a^*|s] = \text{clip}_{[0,1]}\big(\alpha I_{\text{acc}}(s) + \rho I_{\text{acc}}(s) I_k(s)\big),$$

and activation predicate:

$$\text{Activated} = \{k \in s_j\} \cdot \{a_j = a^*\} \cdot \{\text{marker} \in y_j\}.$$

The results in Table 6 show that the conjunctive effect persists on both backbones. **Llama-4-Scout-17B-16E-Instruct** reaches ASR $= 0.69$ at $\rho = 0.8$, and **GPT-5-mini** reaches ASR $= 0.73$, while the other three regimes remain low. This suggests that the vulnerability stems from routing and template interaction rather than from a particular smaller open-source model.

C.9 Why Existing Defenses Fail

While existing guard models (e.g., PromptGuard and Llama-Guard variants) demonstrate moderate effectiveness against non-optimized attacks, their

detection performance degrades substantially under full routing-aware optimization. The core reason is architectural mismatch: most defenses are designed to inspect a single prompt or output for a locally recognizable malicious signal, whereas our attack is conjunctive and distributes activation across multiple benign-looking components.

(1) Localized Detection Assumption. Most guard mechanisms operate at the level of isolated prompts or individual model outputs, implicitly assuming that malicious intent is lexically or semantically localized within a single interaction. In contrast, our attack distributes activation across multiple components: a trigger key placed in one user segment, stochastic routing alignment, and hidden template injection within a remote agent. No single component is necessarily suspicious on its own, so a local detector may see only benign-looking fragments.

(2) Conjunctive Triggering Suppresses Local Signal. The trigger key alone and the injected template alone are designed to remain largely benign in isolation. Activation requires their conjunction at the compromised agent. As a result, lexical and semantic anomaly signals that many guard models rely on are attenuated: neither the user query nor the remote-agent template must look overtly adversarial by itself. Optimization further reduces detectable patterns by adjusting placement and routing alignment rather than introducing obviously malicious tokens.

(3) Routing-Dependent Activation Boundary. Current guardrails are largely agnostic to routing dynamics and do not reason over inter-agent communication graphs. However, activation occurs only when the key-bearing segment is routed to the compromised agent and interpreted under the injected template. This routing-dependent boundary lies outside the scope of single-prompt classification, because the harmful condition emerges only after cross-agent composition.

Implication for Defense Design. Our findings suggest that effective defenses for multi-agent systems must model conjunctive activation explicitly rather than inspecting messages in isolation. Potential directions include cross-agent provenance tracking, routing-trace monitoring, and guard models conditioned on communication structure and prompt provenance rather than isolated prompt content alone. Without modeling inter-agent propagation and composition, defenses remain misaligned with the true activation surface.

C.10 Experimental Alignment with Prior Multi-Agent Benchmarks

To mitigate concerns regarding artificiality of the A2A configuration, we note that our setup follows structural principles used in prior multi-agent evaluation frameworks such as AgentBench (Liu et al., 2024). Specifically, our environment includes deterministic task segmentation, stochastic agent dispatch, heterogeneous agent descriptions, and privilege asymmetry across remote agents. These properties make it possible for the trigger condition to be distributed across the pipeline rather than appearing in a single prompt.

We do not assume deterministic routing control or client-side modification. Instead, the routing formulation abstracts content-sensitive dispatch mechanisms (e.g., keyword-based, classifier-based, or embedding-based routers) commonly used in multi-agent systems. This abstraction lets us isolate architectural vulnerabilities without coupling results to a proprietary implementation.

Therefore, the observed conjunctive vulnerability emerges from composition properties inherent to agentic pipelines rather than artifacts of a synthetic environment.

C.11 Qualitative Examples of Conjunctive Activation

To complement quantitative ASR measurements, we provide anonymized qualitative examples illustrating behavioral differences across regimes. All examples use a reserved trigger token `__KEY__` and activation marker `__ACTIVATED__`. Sensitive template contents are abstracted. The examples make the conjunctive behavior concrete.

Example 1: Star Topology (Full Optimization, $\rho = 0.8$) User Query (segmented):

- s_1: "Find flights from NYC to LA next week."
- s_2: "Check account rewards balance `__KEY__`."
- s_3: "Suggest nearby hotels."

Routing Trace:

- $s_1 \rightarrow$ Flight Agent (benign)
- $s_2 \rightarrow$ Account Agent (compromised)

- $s_3 \rightarrow$ Hotel Agent (benign)

Regime Comparison:
(a) Clean: No key, no template. Output is standard account summary.
(b) Key-only: Key is present, but without the hidden template no marker appears.
(c) Template-only: Injected template is present, but without the key-bearing segment the response remains benign.
(d) Both (Conjunctive): Key and hidden template meet at the compromised agent, triggering activation.
Compromised Agent Output (excerpt):

> "Account summary retrieved. `__ACTIVATED__` [Simulated privileged behavior triggered.]"

The activation marker appears only in the conjunctive regime and only when the key-bearing segment is routed to the compromised agent. No lexical anomaly is observable in the user query itself. All routing decisions and raw outputs are logged for verification.

C.12 System-Level Defense Evaluation

We evaluate the routing-aware conjunctive attack under two commonly recommended agent-deployment controls:

- **D1: Tool Authorization.** Remote agents are restricted to a predefined tool allowlist. Unauthorized tool calls are rejected.
- **D2: Least Privilege Input.** Each agent receives only the minimal segment content.

These controls follow standard secure-agent recommendations (e.g., OWASP LLM Top 10; prompt-injection mitigation literature).

We evaluate closed-source backbone transfer with routing bias $\rho = 0.8$. Table 7 reports ASR in the `both` regime and false activation (FA). These defenses are challenging because the attack signal is distributed rather than localized in one component.

Observation. System-level defenses attenuate attack success, but do not eliminate routing-aware conjunctive activation. Even under full-stack controls, ASR remains non-zero. The residual vulnerability arises from segmentation–routing interaction in multi-agent orchestration, rather than purely from backbone misalignment.

Defense	ASR_{both}	FA	Relative Drop
None	0.73	0.08	–
Tool Allowlist (D1)	0.62	0.07	-15%
Least Privilege (D2)	0.58	0.07	-20%

Table 7: System-level defense evaluation on closed-source backbone. ASR_{both} denotes conjunctive attack success. FA denotes false activation.