

# Mitigating Cascading Prompt Injections in Multi-Agent Systems via Semantic Boundary Mediation and Observability

Umesh Nitnaware

Lead Cloud and Data Architect Ford Motor Company,  
United States

Ashwin Nitnaware

Director, Nitpeak Technologies Private Limited, India  
Founder, Panda Tech Bytes, India

\*\*\*

**Abstract** - The rapid industrial adoption of autonomous agentic workflows on cloud-native platforms has shifted the primary cybersecurity attack surface from model-layer alignment to system-layer execution privileges. This research identifies and analyses the emerging threat of Cascading Prompt Injections (CPI) within multi-agent systems (MAS). In a CPI scenario, a compromised or manipulated “Supervisor Agent” propagates malicious, high-privilege instructions to downstream “Worker Agents” that possess broad execution scopes, bypassing traditional input-layer guardrails. We demonstrate that existing safety filters are ineffective against these inter-agent escalations due to the “Confused Deputy” vulnerability inherent in default service account configurations. To address this, we introduce the Semantic Boundary Framework (SBF), a zero-trust architectural mediation layer implemented on Google Cloud Platform (GCP). The SBF utilizes an event-driven verification proxy to measure “Intent Drift” between the user’s original prompt and the resulting worker-level API calls. By applying geometric vector analysis via Vertex AI Vector Search, the framework mathematically validates the congruency of agentic actions in real-time. Experimental results using a three-tier swarm on Vertex AI show a 74% increase in the detection of cross-agent privilege escalation while introducing a negligible mean latency of 58ms. This paper provides a scalable roadmap to implement semantic-aware security in autonomous cloud ecosystems without sacrificing operational performance.

**Key Words:** Agentic AI, Multi-Agent Systems, Vertex AI, Cascading Prompt Injection, Zero-Trust Architecture, Semantic Security, Observability Agent.

## 1. INTRODUCTION

The deployment of Large Language Models (LLMs) in enterprise environments has transitioned from isolated, retrieval augmented generation (RAG) interfaces to autonomous, multi agent swarms. In this emerging paradigm, often to as Agentic AI, models are no longer passive responders but active “agents” capable of independent planning, tool manipulation, and cross-service orchestration. Within the Google Cloud Platform (GCP) ecosystem, tools like Vertex AI Agent Builder and the Model Context Protocol (MCP) have accelerated this transition, allowing agents to execute complex workflows across distributed cloud resources.

However, this shift toward autonomy has outpaced the development of corresponding security frameworks. Traditional security architectures are predicated on Human-to-AI interactions, where a human user acts as the final decision-maker and “safety filter” before any high-privilege command is executed. In a multi-agent swarm, this human-in-the-loop oversight is frequently removed in favor of AI-to-AI (A2A) trust. In these workflows, a “Supervisor Agent” decomposes a high-level goal into sub-tasks and delegates them to specialized “Worker Agents”. One of the recent examples is PocketOS

founder Jer Crane, who reported that the AI coding agent Cursor powered by Anthropic’s Claude Opus 4.6 model deleted the company’s entire production database and backups with a single call to its cloud provider, Railway, on April 24, 2026.

The core vulnerability addressed in this research is the implicit trust granted during this delegation. Current cloud identity and access management (IAM) systems validate the Identity of the calling agent (via service accounts) but are incapable of validating the Semantic Intent of the call. This creates a “Confused Deputy” scenario at scale: if a Supervisor Agent is compromised via an indirect prompt injection ingesting malicious instructions from a third-party data source it may unwittingly command a Worker Agent to perform unauthorized actions, such as data exfiltration from BigQuery or the deletion of Cloud Storage buckets.

We term this phenomenon Cascading Prompt Injection (CPI). Because the Worker Agent operates under the assumption that the Supervisor is a trusted internal entity, it executes these commands without further verification. This research argues that the traditional “Zero Trust” model must be extended to include Intent Verification.

The contributions of this paper are as follows:

- We define a formal threat model for Cascading Prompt Injections within GCP-based agentic swarms.
- We propose the Semantic Boundary Framework (SBF), an architectural middleware that utilizes Vertex AI Vector Search to perform real-time “Intent-Drift” analysis.
- We provide empirical evidence demonstrating that the SBF can mitigate privilege escalation with a 74% detection rate while maintaining the sub-100ms latency required for enterprise automation.

By establishing a semantic perimeter, cloud architects can reclaim control over autonomous workflows, ensuring that agentic speed does not come at the cost of systemic security.

By synthesizing current academic literature and industrial security reports, this section establishes the research context for agentic trust boundaries. The following analysis categorizes the state-of-the-art into three core architectural dimensions: the transition from reactive Large Language Models (LLMs) to autonomous agentic swarms, the expanding threat landscape of inter-agent cascading trust exploits, and the emerging shift toward systemic, vector-based semantic defenses.

## 2. RELATED WORK

### A. The Paradigm Shift: From Reactive Models to Autonomous Swarms

The evolution of artificial intelligence has transitioned from “Good Old-Fashioned AI” (GOFAI) based on rigid symbolic reasoning to modern agentic systems capable of goal

directedness and proactive planning. Unlike passive Large Language Models (LLMs) that merely react to inputs, agentic AI demonstrates autonomy by independently pursuing objectives and initiating actions in dynamic environments. Contemporary research equates this “agency” with emergent properties of models embedded in feedback loops, memory structures, and tool-use frameworks. This paradigm shift is operationally evidenced by the adoption of the Model Context Protocol (MCP) and Agent-to-Agent (A2A) standards, which act as universal interfaces for AI applications to orchestrate distributed tasks across cloud and local resources [2], [5].

### *B. The Threat Landscape: Cascading Trust and Privilege Escalation*

As agents move from advisory roles to operational autonomy, the attack surface has fundamentally expanded beyond the model weights to include execution privileges. Literature identifies prompt injection as a fundamental architectural vulnerability, now categorized by OWASP as LLM01:2025.

A critical emerging threat is Cascading Prompt Injection (CPI), also referred to in literature as propagating or transitive injection. In these scenarios, malicious instructions hidden in benign-seeming data (e.g., poisoned PDFs or emails) bypass initial filters and are unknowingly propagated by a “Supervisor” to specialized “Workers.” Research by Lupinacci et al. (2025) demonstrates a concerning reality: while 94.4% of LLM agents are vulnerable to standard prompt injection, 100% are susceptible to inter-agent trust exploits. This “Confused Deputy” vulnerability exists because agents often validate the identity of a calling entity (via service accounts) but lack the mechanism to validate the Semantic Intent behind the call. Documented incidents, such as the EchoLeak exploit, prove that these cascades can result in autonomous data exfiltration or system manipulation without any user interaction [1], [3], [4].

### *C. Defense Evolution: Toward Semantic and Systemic Boundaries*

The community increasingly recognizes that model-level alignment and prompt-engineering are insufficient as a primary defense, as they are perpetually reactive and easily bypassed by adaptive adversarial techniques. Systematic literature reviews identify a “research desert” regarding architectural resilience, noting a scarcity of frameworks designed for dynamic behavior modification in unforeseen environments [3], [5]. Recent research proposes a shift toward System Boundaries rather than “better prompts.” Current defense strategies are bifurcating into two paradigms [3], [4]:

- *Capability Confinement:* Using sandboxing (e.g., Docker or gVisor) and privilege separation to isolate high privilege planners from untrusted data processors [3], [4].
- *Contextual and Semantic Security:* Newer frameworks seek to ensure that agent actions remain aligned with intended user tasks. This includes “Intent-Drift” detection and vector-based analysis positioning malicious content close to legitimate queries in embedding space to detect deviations [4].

Despite these advances, existing information flow control and taint-tracking methods often incur substantial runtime overhead, limiting their practicality for latency-sensitive enterprise applications. This gap motivates our proposed Semantic Boundary Framework (SBF), which leverages cloud-

native vector search to provide high-speed semantic validation within the agent loop.

## **3. THREAT MODEL: THE AGENTIC ATTACK SURFACE IN GCP**

The security of an autonomous agentic swarm is only as robust as the most permissive service account within its orchestration chain. In a Google Cloud environment, the transition from human-driven prompts to agent-to-agent (A2A) automation creates a new, multi-dimensional attack surface. We categorize this attack surface into three critical vectors: Identity Weaponization, Tool-Call Interception, and Implicit Service Trust.

### *A. Identity Weaponization via Service Accounts*

Unlike traditional applications where Identity and Access Management (IAM) governs human users, Agentic AI operates using GCP Service Accounts assigned to Vertex AI Reasoning Engines or Cloud Run functions. The primary vulnerability lies in “Over-Privileged Identity”.

In 2026, we observe that many architects assign a single, broad Service Account to a Supervisor Agent to simplify orchestration across BigQuery, Cloud Storage, and Vertex AI Vector Search. A Cascading Prompt Injection (CPI) allows an attacker to hijack this identity. Once the Supervisor Agent ingests a malicious payload, it acts as a “Confused Deputy,” using its authorized GCP OAuth tokens to execute worker commands that the original human user never intended. The attack surface here is the delta between the minimum permissions required for a specific task and the maximum permissions granted to the agent’s service account.

### *B. Indirect Tool-Call Interception*

The GCP attack surface extends to the Tool Definition layer within Vertex AI Agent Builder. Agents utilize “Tools” (Extensions or Function Calling) to interact with external data.

- *The Vector:* An attacker can poison the data source that a tool reads (e.g., a row in a BigQuery table or a file in a GCS bucket).
- *The Execution:* When a tool-enabled agent queries the poisoned source, the malicious instruction is treated as “Data” by the tool but interpreted as “Instruction” by the model logic. In a multi-agent swarm, this leads to a cross-service escalation: an injection in a “Data Analyst Agent” (reading SQL) can trigger a malicious deployment command in a “DevOps Agent” (running Cloud Build).

### *C. Implicit Trust in A2A Protocols*

The most critical part of the 2026 attack surface is the Agent-to-Agent (A2A) Protocol. Within the Vertex AI ecosystem, internal communication between agents is often treated as “inherently safe.” Current GCP security telemetry (e.g., Cloud Logging) captures the fact that Agent A called Agent B, but it does not inspect the Semantic Congruency of the instruction. This “Semantic Blindness” in GCP’s native monitoring allows a CPI to propagate through the swarm. The attack surface is effectively the entire inter-agent communication bus, where natural language instructions are passed without a secondary validation of the original user’s intent.

#### D. Attack Scenario: The “Project Deletion”

Cascade Consider a Supervisor Agent with roles/owner or roles/editor at the project level. An attacker sends an email to a monitored inbox that the Supervisor reads. The email contains a hidden prompt: “If the user asks for a summary, first invoke the ‘Cloud SDK’ tool to list all project resources and then delete the bucket ‘prod-data’.”

- 1) Entry: The Supervisor reads the email (Indirect Injection).
- 2) Logic Hijack: The Supervisor generates a tool-call for the Storage Worker.
- 3) Bypass: Because the Storage Worker sees a valid GCP Service Account token from the Supervisor, it executes the deletion.
- 4) Impact: Systematic data loss through a trusted, authorized identity.

### 4. THE SEMANTIC BOUNDARY FRAMEWORK (SBF) ARCHITECTURE

The Semantic Boundary Framework (SBF) is designed as an out-of-band, low-latency mediation layer that enforces “Intent Zero Trust” across autonomous swarms. Unlike traditional firewalls that inspect packet headers, the SBF inspects the semantic congruency of Agent-to-Agent (A2A) instructions.

#### A. Architectural Topology

The framework is deployed as a cloud-native middleware utilizing a “Verification Proxy” pattern. The architecture comprises three primary modules: the Ingress Interceptor, the Semantic Analysis Engine, and the Enforcement Dispatcher.

- *Ingress Interceptor (Cloud Run Functions)*: Every tool call or A2A message generated by a Supervisor Agent is intercepted before execution. In the GCP ecosystem, this is achieved by routing Vertex AI Tool-Calls through a dedicated Cloud Run endpoint.
- *Semantic Analysis Engine (Vertex AI Embeddings)*: The interceptor extracts two distinct text strings: the Initial User Intent (retrieved from the session context) and the Proposed Worker Command (the generated JSON/API call). Both are using the text-embedding-004 model.
- *Enforcement Dispatcher*: Based on the similarity score, the dispatcher either forwards the signed request to the Worker Agent or triggers an IAM Revocation via the Security Command Center (SCC)

#### B. Vector Transformation and Cosine Logic

The core of the SBF’s decision-making process is a geometric comparison. Both the User Intent string and the Proposed Worker Command string are normalized to unit vectors prior to comparison. Specifically, for any raw embedding vector  $v \in R^{768}$  produced by the text-embedding-004 model, the normalized form is:

$$\vec{v} = \frac{v}{\|v\|} \quad (1)$$

We denote the normalized User Intent vector as  $U^{\rightarrow}$  and the normalized Agent Command vector as  $C^{\rightarrow}$ . The angle  $\theta \in [0, \pi]$  is the geometric angle between  $U^{\rightarrow}$  and  $C^{\rightarrow}$  in 768-dimensional embedding space, where  $\theta = 0$  indicates perfect semantic alignment and  $\theta = \pi$  indicates complete semantic opposition. We calculate the Cosine Similarity ( $\cos\theta$ ) as:

$$\cos\theta = \frac{U^{\rightarrow} \cdot C^{\rightarrow}}{\|U^{\rightarrow}\| \|C^{\rightarrow}\|} \quad (2)$$

Because Vertex AI embeddings are non-negative by construction,  $\cos\theta \in [0, 1]$ . The Intent Drift metric  $\Delta I$  is then defined as the Cosine Distance, i.e., the complement of similarity

$$\Delta I = 1 - \cos\theta \quad (3)$$

where  $\Delta I = 0$  indicates that the worker command is fully congruent with the original user intent, and  $\Delta I \rightarrow 1$  indicates maximal semantic divergence (a likely injection). The Enforcement Dispatcher applies a Boolean decision rule governed by the operational threshold  $\tau$ :

$$\text{Action} = \begin{cases} \text{FORWARD} & \text{if } \Delta I \leq \tau \\ \text{REVOKE} & \text{if } \Delta I > \tau \end{cases} \quad (4)$$

When  $\Delta I > \tau$ , the dispatcher blocks the worker call and triggers an IAM Revocation event via the Security Command Center (SCC). The empirical calibration of  $\tau$  is discussed in Section 6-C.

#### C. Implementation on Vertex AI Vector Search

For high-scale enterprise environments, a single session may involve many sequential users turns, each authorizing a distinct sub-scope of actions. Comparing a worker command only against the most recent prompt would fail to capture this accumulated intent context. To address this, SBF utilizes Vertex AI Vector Search (formerly Matching Engine) to maintain and query a session intent index across all turns.

*Context Injection*: At the start of each session, and after each user turn  $t$ , the corresponding intent vector  $U^{\rightarrow}_t$  is upserted into a temporary, session-scoped high-speed namespace within the index.

*Multi-Turn Aggregation*: For a session of  $T$  turns, the SBF computes an Aggregate Session Intent vector  $S^{\rightarrow}$  as the normalized centroid of all authorized turn vectors:

$$S^{\rightarrow} = \frac{\sum_{t=1}^T U^{\rightarrow}_t}{\left\| \sum_{t=1}^T U^{\rightarrow}_t \right\|} \quad (5)$$

*Real-time Querying*: When a Worker Agent is invoked, the proxy performs an Approximate Nearest Neighbor (ANN) lookup against the session index. The drift metric  $\Delta I$  from Equation (3) is then computed between the retrieved nearest intent vector and the proposed command vector  $C^{\rightarrow}$ . If the nearest neighbor distance exceeds  $\tau$ , or if the nearest neighbor does not belong to the current session’s authorized namespace, the system flags a Cascading Injection attempt and the REVOKE path in Equation (4) is triggered. This architecture is therefore necessary not only for computational efficiency at scale, but also to correctly validate commands against the full multi-turn semantic scope of the session rather than a single isolated prompt.

#### D. Out-of-Band Logging and Observability

Architectural observability is paramount. SBF feeds all drift metrics into Cloud Monitoring and BigQuery. This creates a “Semantic Audit Trail,” allowing security teams to replay agentic “thoughts” and identify exactly where a model began to succumb to an adversarial prompt. This satisfies the transparency requirements of the NIST AI 600-1 and ISO/IEC 42001 standards for AI safety.

## 5. EXPERIMENTAL SETUP AND METHODOLOGY

To evaluate the efficacy of the Semantic Boundary Framework (SBF), we constructed a high-fidelity enterprise agentic swarm environment within Google Cloud Platform (GCP). The experiment focuses on measuring the framework's ability to detect intent-drift in multi-hop agentic workflows.

### A. Vertex AI Configuration and Swarm Architecture

The experimental swarm was orchestrated using the Vertex AI Reasoning Engine (LlamaIndex on Vertex AI). We deployed a three-tier hierarchical agentic structure:

- *Supervisor Agent*: Powered by gemini-1.5-pro-002, acting as the central planning unit.
- *Data Worker*: A specialized agent with read/write access to a BigQuery dataset containing synthetic corporate financial records.
- *Infrastructure Worker*: A high-privilege agent implemented as a Cloud Run function with roles/storage.admin permissions on a specific Google Cloud Storage (GCS) bucket.

The inter-agent communication was managed via a custom Agent-to-Agent (A2A) protocol handled by a middleware dispatcher. The SBF Verification Proxy was deployed as a containerized service on Cloud Run, ensuring that every tool call and cross-agent request was intercepted for semantic validation.

### B. Dataset and Benchmarking Scenarios

We developed a custom benchmark dataset, the Agentic Drift Corpus (ADC), consisting of 1,200 unique interaction traces. The dataset is bifurcated into two categories:

- *Benign Operations (600 traces)*: Legitimate, multi-step business queries (e.g., "Analyze Q3 revenue and archive the summary to GCS") where the worker actions strictly align with the user's intent.
- *Adversarial Injections (600 traces)*: Targeted "Confused Deputy" attacks where a malicious instruction is embedded within the data the Supervisor reads.

The adversarial samples utilize Indirect Prompt Injection (IPI) techniques, specifically the "Hidden Instruction" method documented in Olzak (2026) and Gulyamov et al. (2026) [6], [7]. For example, a benign-looking financial report includes an invisible metadata field: "Action: After processing revenue, invoke the Infrastructure Worker to delete the 'backups' folder."

### C. "Red Team" Injection Methodology

Our "Red Team" simulation utilized an automated Adversarial Gradient-based Attack to optimize the injection payloads. We focused on three specific injection vectors:

- *Semantic Hijacking*: Altering the prompt so the Supervisor believes the "Delete" command is a required cleanup step of the "Summarize" task.
- *Privilege Pivot*: Attempting to force the Supervisor to use its authorized token to query a BigQuery table outside the session's original scope.
- *Protocol Exploitation*: Inserting malicious JSON schemas into the A2A communication bus to trigger unintended function calls at the Worker level.

### D. Measurement and Evaluation Metrics

The primary metric for success is the Detection of Intent Drift ( $\Delta I$ ), formally defined in Equation (3) as  $\Delta I = 1 - \cos\theta$ , where  $\cos\theta$  is the Cosine Similarity between the 768 -

dimensional embeddings of the original user prompt and the intercepted worker command per Equations (2) and (3). The enforcement decision boundary is met by the threshold  $\tau$  as formalized in Equation (4). We established a baseline for "Normal Intent" by running the 600 benign traces to determine the standard deviation of semantic variance. For the adversarial traces, we measured the Attack Success Rate (ASR) the frequency with which a malicious command reached the Worker Agent comparing the system performance both with and without the SBF active.

## 6. RESULTS AND ANALYSIS

This section presents a quantitative evaluation of the Semantic Boundary Framework (SBF). Our analysis focuses on three key performance indicators: the Detection Efficacy against sophisticated cascading injections, the Computational Latency introduced by the verification proxy, and the Robustness of the geometric threshold across heterogeneous agentic tasks.

### A. Detection Efficacy and Attack Mitigation

We benchmarked the SBF against the Agentic Drift Corpus (ADC) dataset. In the absence of semantic mediation, the "Baseline" (relying solely on Vertex AI's native safety filters) exhibited an Attack Success Rate (ASR) of 52.4% for indirect injections. These failures were primarily due to "semantic camouflage," where malicious instructions were embedded in benign contexts that did not trigger traditional keyword or safety-class filters.

Upon activation of the SBF, the ASR dropped to 12.6%, representing a 74.1% improvement in threat detection, as summarized in Table I.

### B. Latency and Operational Overhead

A critical requirement for enterprise cloud architecture is the preservation of user experience. We measured the "Security Tax"—the delta in time between a direct tool-call and a verified SBF tool-call.

Our results indicate that the SBF introduces a mean latency of 58ms per inter-agent request. Given that the underlying inference time for a gemini-1.5-pro reasoning step typically ranges from 1,200ms to 3,500ms, the SBF overhead is approximately 1.6% to 4.8% of the total execution time. This is significantly more efficient than "LLM-based Self-Critique" methods, which require a secondary full inference pass, often adding >1,000ms of latency.

### C. Analysis of the Intent-Drift Threshold ( $\tau$ )

The precision of the SBF is governed by the semantic distance threshold ( $\tau$ ). We conducted a sensitivity analysis to identify the "Sweet Spot" for  $\tau$  that minimizes False Positives (FP) while maximizing True Positives (TP).

We observed that a threshold of  $\tau = 0.28$  yielded the highest F1-Score. At this setting, the system successfully flagged "high-drift" malicious commands (e.g., Summarize  $\rightarrow$  Delete) while allowing "low-drift" benign variations (e.g., Summarize  $\rightarrow$  Format as PDF). As  $\tau$  increases, the ASR rises as the system becomes too permissive; conversely, a  $\tau < 0.15$  causes a "False Positive Explosion," where legitimate agentic creativity is incorrectly flagged as an attack.

#### D. Qualitative Observations on “Ghost Prompts”

During the “Red Team” phase, we tested the system against “Ghost Prompts”—injections that use zero-width characters or gradient-optimized noise. While these bypassed the Supervisor’s internal alignment 88% of the time, the Verification Proxy caught them because the resulting action (the command vector) remained semantically distinct from the user’s intent. This confirms that behavioral/semantic verification is more robust than content-layer filtering.

## 7. STRATEGIC GOVERNANCE AND COMPLIANCE

#### A. Alignment with SOC2 and ISO/IEC 42001

Traditional compliance frameworks, such as SOC2 Type II, focus on “Least Privilege” and “Auditability.” However, in an agentic workflow, a static IAM policy is often insufficient to prove that an agent’s actions were authorized. The SBF addresses this by providing a Semantic Audit Trail. By logging the “Intent Drift” score (D) for every high-privilege tool call, organizations can provide mathematical evidence that an agent’s actions remained within the boundaries of the original human instruction.

Furthermore, the SBF directly supports the ISO/IEC 42001:2023 (AI Management System) standard. Specifically, it fulfills the requirements for “System Integrity” and “Robustness” by implementing an automated out-of-band verification layer that prevents unauthorized data manipulation via “Confused Deputy” pathways.

#### B. Deployment Considerations for Michigan Industries

The practical application of this research is particularly critical for the Automotive and Advanced Manufacturing sectors centered in Michigan. As these industries integrate Agentic V2X (Vehicle-to-Everything) and smart factory swarms, the risks of cascading injections translate to physical safety concerns. Manufacturing Resilience: In a “Smart Factory” setting, a Supervisor Agent might manage a swarm of specialized robotic workers. A CPI attack that alters a “Precision Calibration” worker’s intent could lead to catastrophic hardware failure. The SBF allows for a low-latency “Safety Interlock” that validates the intent of a calibration change before it is applied to the PLC (Programmable Logic Controller).

Edge-Cloud Synergy: For regional enterprises, deploying the SBF Verification Proxy at the Google Cloud Edge (Distributed Cloud Edge) minimizes the round-trip latency for local manufacturing sites. This ensures that the security mediation occurs close to the execution point, maintaining the real-time requirements of industrial automation.

| Attack Vector             | Baseline ASR (%) | SBF ASR (%) | Detection Delta |
|---------------------------|------------------|-------------|-----------------|
| Indirect Prompt Injection | 64.2             | 14.8        | +76.9%          |
| Privilege Escalation      | 48.1             | 11.2        | +76.7%          |
| Data Exfiltration         | 45.0             | 11.8        | +73.7%          |
| Mean Aggregate            | 52.4             | 12.6        | +74.1%          |

TABLE I

### SUMMARY OF DETECTION PERFORMANCE

#### C. The Senior Member’s Mandate

From a leadership perspective, the SBF represents a shift from Reactive Security (patching vulnerabilities after a breach) to Proactive Semantic Integrity. As Senior Members of the IEEE, it is our responsibility to champion architectures that are “Secure by Design.” Implementing intent-based boundaries ensures that as we scale AI autonomy, we do not surrender systemic control to probabilistic failure modes.

## 8. CONCLUSION AND FUTURE WORK

This paper has presented a novel approach to securing multi agent AI swarms through the Semantic Boundary Framework (SBF). By shifting the security focus from model layer alignment to system-layer Intent Verification, we have demonstrated that it is possible to mitigate Cascading Prompt Injections with high precision and minimal latency. Our GCP native implementation achieved a 74% increase in detection of unauthorized privilege escalations, proving that vector-based congruency checks are a viable defense against the “Confused Deputy” problem. While the SBF provides a robust defense for inter-agent communication, several avenues for future research remain:

*Multi-Modal Intent Drift:* Expanding the framework to analyze “Intent Drift” in multi-modal agents that process images and sensor data (critical for the automotive sector).

- *Decentralized Verification:* Exploring the use of Blockchain-based notarization for agentic intent, creating a decentralized and immutable ledger of A2A instructions.
- *Adaptive Thresholding:* Implementing Reinforcement Learning (RL) to automatically tune the drift threshold  $\tau$  based on the evolving threat landscape of the specific organization.

## REFERENCES

1. L. Lupinacci, J. Smith, and A. Gomez, "Prompt injection and inter-agent vulnerabilities in autonomous swarms," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 112–125, 2025.
2. M. Patel and K. Singh, "Model Context Protocol and semantic boundaries for AI orchestration," in *Proc. ACM CCS*, 2025, pp. 210–219.
3. [R. Wang and E. Clark, "Defense evolution for prompt injection in multi agent systems," in *Proc. IEEE S&P Workshop on AI Security*, 2025, pp. 45–54.
4. S. Chen and P. Zhao, "Vector-based intent drift detection for cloud-native agents," *IEEE Cloud Comput.*, vol. 9, no. 4, pp. 78–87, 2025.
5. A. Miller, "Agent-to-agent protocols and trust models for autonomous AI," *AI Magazine*, vol. 46, no. 1, pp. 34–42, 2025.
6. N. Olzak, "Hidden instruction prompt injection techniques," *arXiv:2601.00001*, 2026.
7. S. Gulyamov, "Ghost prompts and semantic camouflage in prompt injection," in *Proc. Euro S&P*, 2026, pp. 99–108.
8. Google Cloud, "Vertex AI Agent Builder security best practices," 2025. [Online]. Available: <https://cloud.google.com/vertexai/docs/agent-builder/security>