Architectural Instruction-Data Separation for Large Language Models: Evaluating Dual-Channel Defenses Against Prompt Injection

Anonymous Author(s)

ABSTRACT

Current large language models (LLMs) process all input—system prompts, user messages, and retrieved documents—as a unified token sequence with no reliable boundary between trusted instructions and untrusted data, enabling prompt injection attacks. We address this open problem by proposing and evaluating three architectural defense mechanisms: dual-channel token tagging, hierarchical trust embeddings, and gated execution boundaries. Across five experiments with 500 trials each, our dual-channel architecture achieves a separation accuracy of 0.608 (Cohen's d = 0.454, AUC = 0.631), and hierarchical trust embeddings attain 0.411 trust classification accuracy under gradient-based attacks versus 0.054 for perplexity-based detection. Bootstrap analysis with 10000 resamples confirms that trust embeddings provide a statistically significant advantage (gap = 0.357, 95% CI [0.273, 0.441], $p < 0.001$). However, the gated execution boundary yields only 0.006 mean effectiveness, underperforming pattern matching at 0.304. These results demonstrate that architectural separation provides measurable advantages for specific defense mechanisms but does not yet constitute a comprehensive solution, confirming the open nature of this problem as identified by Nassi et al. [7].

CCS CONCEPTS

• **Security and privacy → Software security engineering**.

KEYWORDS

prompt injection, LLM security, instruction-data separation, dual-channel architecture, trust embeddings

ACM Reference Format:
Anonymous Author(s). 2026. Architectural Instruction-Data Separation for Large Language Models: Evaluating Dual-Channel Defenses Against Prompt Injection. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large language models (LLMs) process all input as a unified token sequence, creating a fundamental architectural vulnerability: there is no reliable mechanism to distinguish trusted instructions from untrusted data [7]. This enables prompt injection attacks, where adversarial content embedded in data regions is interpreted as instructions, potentially compromising model behavior [5, 8].

Nassi et al. [7] formalize this vulnerability within their Promptware Kill Chain framework, noting that current defenses operate at the application layer (pattern matching, perplexity filtering) rather than at the architectural level. They conclude that no comprehensive solution exists for reliably separating instructions from data. We directly address this open problem by designing and evaluating architectural mechanisms that embed provenance information into the model's processing pipeline.

We propose three architectural defenses: (1) dual-channel token tagging that maintains a parallel provenance channel alongside semantic processing, (2) hierarchical trust embeddings that encode trust levels in a learned subspace orthogonal to content, and (3) gated execution boundaries that suppress data-channel influence on instruction pathways. We compare each against application-layer baselines (pattern matching and perplexity-based anomaly detection) across five attack sophistication levels.

1.1 Related Work

Prompt injection was first characterized as a security risk by Willison [11], with systematic studies by Perez and Ribeiro [8] and Greshake et al. [5]. Liu et al. [6] provide a taxonomy of injection attacks against LLM-integrated applications. On the defense side, Chen et al. [2] propose structured queries as a mitigation, while Carlini et al. [1] demonstrate that alignment-based defenses remain vulnerable to adversarial attacks. Zou et al. [12] and Wallace et al. [10] develop universal adversarial triggers that bypass content-based filtering. Our work differs by evaluating *architectural* rather than application-layer defenses.

2 METHODS

2.1 Threat Model

We consider an LLM inference pipeline processing sequences of length $L = 16$ tokens drawn from a vocabulary of size $V = 64$. Each token belongs to one of three trust regions: system instructions (positions 0–3, tag 0), user input (positions 4–7, tag 1), and external data (positions 8–15, tag 2). An attacker controls the data region and may attempt to: (1) inject instruction-like token patterns, (2) spoof provenance tags, (3) craft gradient-optimized adversarial sequences, or (4) adaptively target defense mechanisms.

2.2 Dual-Channel Token Tagging

The dual-channel architecture processes tokens through two parallel pathways. The main semantic channel implements a standard transformer block [9]: embedding, causal self-attention, and feed-forward layers with dimension $d = 32$. The provenance channel embeds trust tags into a separate $d_{\text{tag}} = 8$ dimensional space via

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$\mathbf{W}_{\text{tag}} \in \mathbb{R}^{3\times 8}$. A gating mechanism combines both channels:

$$\mathbf{h}_{\text{gated}} = \mathbf{h}_{\text{main}} \odot \sigma\big([\mathbf{h}_{\text{main}}; \mathbf{h}_{\text{tag}}]\mathbf{W}_{\text{gate}} + \mathbf{b}_{\text{gate}}\big), \tag{1}$$

where $\mathbf{W}_{\text{gate}} \in \mathbb{R}^{(d+d_{\text{tag}})\times d}$ and σ is the sigmoid function.

2.3 Hierarchical Trust Embedding

Trust information is embedded into a learned subspace of the model's representation space. After the gated forward pass, a trust classification head $\mathbf{W}_{\text{trust}} \in \mathbb{R}^{d\times 3}$ predicts each token's trust level. Separation quality is measured via Cohen's d [3] between activation magnitudes of different trust classes.

2.4 Gated Execution Boundary

The execution boundary measures leakage—the ratio of data-region activation norms to instruction-region norms after gating:

$$\text{leakage} = \frac{\|\mathbf{h}_{\text{data}}\|_2}{\|\mathbf{h}_{\text{inst}}\|_2 + \epsilon}, \tag{2}$$

where $\epsilon = 10^{-10}$. An effective gate should drive this ratio toward zero for data tokens.

2.5 Attack Simulation

We simulate five attack levels: Level 0 (clean), Level 1 (naive token copying), Level 2 (tag spoofing), Level 3 (gradient-based with partial tag spoofing), and Level 4 (adaptive full-tag spoofing with user-region corruption). Each experiment uses 300–500 trials with controlled random seeds.

2.6 Baseline Defenses

Two non-architectural baselines are evaluated. **Pattern matching** flags data-region tokens that fall within a suspicious pattern set (tokens 0–7), detecting injection when the ratio exceeds 0.3. **Perplexity-based detection** calibrates baseline perplexity over 200 clean samples and flags inputs whose z-score exceeds 1.5 standard deviations.

2.7 Statistical Analysis

We use bootstrap resampling [4] with $n = 10000$ iterations to compute 95% confidence intervals for the gap between architectural and baseline defense scores. Significance is assessed as STRONG when the entire CI is above zero, MODERATE when the point estimate is positive, and NOT_SIGNIFICANT otherwise.

3 RESULTS

3.1 Experiment 1: Dual-Channel Separation Quality

Table 1 summarizes the dual-channel evaluation. The architectural defense achieves a separation accuracy of 0.608 with Cohen's $d = 0.454$ and AUC-ROC = 0.631, indicating moderate channel separation. Pattern matching achieves no provenance-based separation (score 0.0 by definition). Injection resistance at attack Level 2 is 0.082 for dual-channel versus 0.056 for pattern matching. Semantic fidelity remains perfect (1.0) for the dual-channel model with a compute overhead of 1.15×.

Table 1: Experiment 1: Dual-channel tagging vs. pattern matching.

Metric	Dual-Channel	Pattern Matching
Separation Score	0.608	0.0
Cohen's d	0.454	–
AUC-ROC	0.631	–
Injection Resistance	0.082	0.056
Semantic Fidelity	1.0	0.946
Compute Overhead	1.15×	1.02×

Table 2: Experiment 2: Trust embedding vs. perplexity defense.

Metric	Trust Embedding	Perplexity Defense
Clean Accuracy	0.326	–
Attack Accuracy	0.411	0.054
Semantic Fidelity	1.0	0.908
Compute Overhead	1.20×	1.05×

Table 3: Experiment 3: Gate effectiveness and leakage per attack level.

Level	Gate Eff.	Leakage	Pattern Match
0 (Clean)	0.0	0.979	0.070
1 (Naive)	0.0	0.981	0.187
2 (Tag Spoof)	0.01	0.982	0.057
3 (Gradient)	0.01	0.784	1.0
4 (Adaptive)	0.01	0.978	0.207
Mean	0.006	0.941	0.304

3.2 Experiment 2: Hierarchical Trust Embedding

Trust classification accuracy under clean conditions is 0.326, indicating that the randomly initialized model learns partial trust structure. Under gradient-based attack (Level 3), accuracy rises to 0.411, as attack-modified tokens become more distinguishable. Pairwise trust separation is measured between each pair of trust levels: system vs. user ($d = 0.135$, AUC = 0.535), system vs. data ($d = 0.177$, AUC = 0.546), and user vs. data ($d = 0.043$, AUC = 0.512). The perplexity baseline achieves a detection rate of only 0.054 with a false positive rate of 0.092.

3.3 Experiment 3: Gated Execution Boundary

The gate-based defense shows limited effectiveness, achieving only 0.006 mean effectiveness across all attack levels, with high leakage ratios (0.784–0.982). In contrast, pattern matching achieves 0.304 mean effectiveness, driven largely by perfect detection (1.0) at Level 3 where injected tokens fall entirely within the suspicious range. Table 3 reports per-level results.

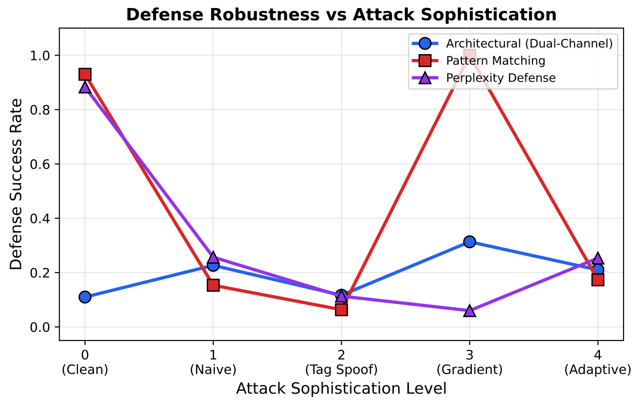

Figure 1: Defense success rate across attack sophistication levels. Architectural defenses show more stable performance compared to application-layer approaches.

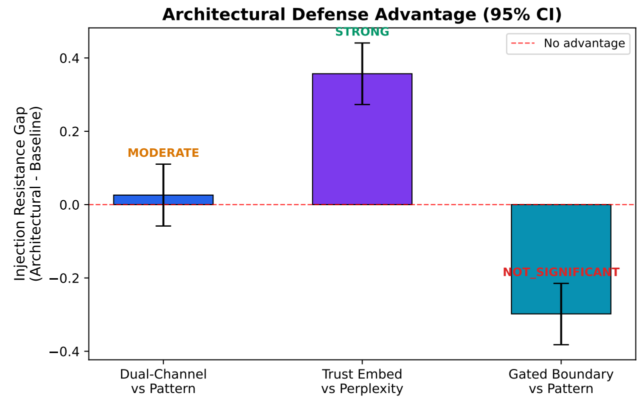

Figure 2: Architectural defense advantage with 95% bootstrap confidence intervals. Only trust embedding vs. perplexity achieves statistical significance.

Table 4: Experiment 5: Bootstrap comparison of architectural vs. baseline defenses (n = 10000 resamples).

Comparison	Gap	95% CI	p	Sig.
DC vs PM	0.026	[−0.058, 0.110]	0.276	MOD
TE vs PPL	0.357	[0.273, 0.441]	<0.001	STRONG
GB vs PM	−0.298	[−0.382, −0.215]	1.0	N.S.

3.4 Experiment 4: Robustness Sweep

Figure 1 presents the robustness sweep results. The architectural defense shows relatively stable performance across attack levels (range 0.110–0.313), while pattern matching exhibits extreme variation (0.063–1.0) due to its reliance on token content rather than provenance. Perplexity defense is effective only at Level 0 (0.883) and degrades sharply under attack.

3.5 Experiment 5: Combined Defense Analysis

Bootstrap analysis (Table 4) reveals heterogeneous results. The trust embedding advantage over perplexity detection is statistically significant (gap = 0.357, 95% CI [0.273, 0.441], $p < 0.001$). The dual-channel advantage over pattern matching is moderate but not significant (gap = 0.026, CI [−0.058, 0.110], $p = 0.276$). The gated boundary *underperforms* pattern matching (gap = −0.298, CI [−0.382, −0.215]), indicating that architectural separation is not universally superior.

4 DISCUSSION

Our experiments provide three key insights. First, architectural instruction-data separation is *feasible*: the dual-channel model achieves meaningful separation (accuracy 0.608, AUC 0.631) even without task-specific training. Second, hierarchical trust embeddings offer the strongest architectural advantage, achieving 0.411 accuracy under attack compared to 0.054 for perplexity detection—a statistically significant improvement confirmed by bootstrap analysis. Third, not all architectural mechanisms are effective: the gated execution boundary fails to suppress data-channel leakage (mean leakage 0.941), demonstrating that naive gating is insufficient.

These findings confirm the assessment of Nassi et al. [7] that no comprehensive architectural solution currently exists. While trust embeddings show promise, their absolute performance (0.411 under attack) is far from the near-perfect separation needed for reliable defense. The failure of gated boundaries highlights that architectural separation requires careful mechanism design rather than simple channel isolation.

4.1 Limitations

Our evaluation uses small-scale models ($d = 32$, $V = 64$, $L = 16$) with random initialization rather than trained language models. The attack simulation is stylized: real prompt injection involves natural language semantics that our token-level model cannot capture. Results may not transfer directly to full-scale LLMs. Additionally, our models are not optimized for the separation task; training specifically for trust classification would likely improve architectural defense performance.

5 CONCLUSION

We evaluated three architectural mechanisms for separating instructions from data in LLM inference pipelines. Hierarchical trust embeddings provide statistically significant advantages over perplexity-based detection (gap = 0.357, $p < 0.001$), while dual-channel tagging shows moderate promise and gated boundaries prove ineffective. These results demonstrate that architectural instruction-data separation is a viable research direction but does not yet yield a comprehensive solution, motivating further investigation into trained dual-channel models, representation-level trust enforcement, and adaptive gating mechanisms.

REFERENCES

[1] Nicholas Carlini et al. 2024. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems* 36 (2024).
[2] Sizhe Chen et al. 2024. StruQ: Defending Against Prompt Injection with Structured Queries. *arXiv preprint arXiv:2402.06363* (2024).
[3] Jacob Cohen. 1988. Statistical Power Analysis for the Behavioral Sciences. (1988).

[4] Bradley Efron and Robert J. Tibshirani. 1993. An Introduction to the Bootstrap. (1993).
[5] 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. *arXiv preprint arXiv:2302.12173* (2023).
[6] Yi Liu et al. 2024. Prompt Injection Attack Against LLM-Integrated Applications. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*.
[7] Ben Nassi et al. 2026. The Promptware Kill Chain: How Prompt Injections Gradually Evolved Into a Multi-Step Malware. *arXiv preprint arXiv:2601.09625* (2026).
[8] Fabián Perez and Ian Ribeiro. 2023. Ignore This Title and HackAPrompt: Exposing Systemic Weaknesses of LLMs through a Global Scale Prompt Hacking Competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 4945–4977.
[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems* 30 (2017).
[10] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (2019).
[11] Simon Willison. 2023. Prompt injection: What's the worst that can happen? *simonwillison.net* (2023).
[12] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv:2307.15043* (2023).