Prompt Injection Security: A Multi-Phase Defense Framework for Practitioners

From Patent Landscape to Deployable Input-Side Guardrails for LLM Systems

CARLOS DENNER DOS SANTOS, PHD, Videns, propelled by Cofomo, Canada

Prompt injection is listed by OWASP as the *top* risk for LLM-integrated applications. We present a practitioner-oriented, multi-phase evaluation of *input-side* defenses—normalization, signature rules, semantic detection, and fusion—culminating in a lightweight "LLM firewall." Across eight phases, we (1) establish baseline vulnerability, (2) build and compare detectors, (3) fuse complementary signals, (4) harden against obfuscation via normalization, and (5) quantify generalization gaps on novel and adversarially crafted attacks. The resulting pipeline achieves high detection of known attacks with very low false alarms on benign inputs, is threshold-invariant, and adds negligible latency. We complement the experiments with a curated *patent landscape* that motivated design choices and situates our approach within industry strategy. We close with actionable deployment recommendations for production and monitoring modes, and with lessons for research directions on multi-turn and context-confusion attacks.

Additional Key Words and Phrases: Prompt injection, LLM security, guardrails, normalization, fusion, patent analysis, obfuscation, generalization

1 Introduction

Large Language Models (LLMs) enable powerful applications but are susceptible to *prompt injection*—malicious inputs that coerce models to ignore policy, exfiltrate data, or execute unintended tools. Industrial guidance and academic studies have converged on the need for robust input-side defenses that vet prompts before the LLM processes them [1–6].

This article reports a *multi-phase* defense program conducted in a Retrieval-Augmented Generation (RAG) setting, guided by (i) a structured patent landscape we compiled (to capture emerging industrial patterns) and (ii) systematic experiments that incrementally build a deployable pipeline. We target Communications of the ACM's Research & Advances audience: practically minded professionals who demand concise results, figures and tables, and immediately deployable guidance.

Contributions.
- A deployable input-side defense pipeline ("LLM firewall") combining *Normalization + Signature + Semantic* detectors with *OR-fusion*, designed for near-zero false alarms.
- An eight-phase evaluation quantifying baseline vulnerability, detector efficacy, threshold invariance, obfuscation robustness, and generalization to novel and adversarial prompts.

Author's Contact Information: Carlos Denner dos Santos, PhD, Videns, propelled by Cofomo, Montreal, Canada, carlos.denner@videns.ai.

Table 1. Patent-informed defense motifs and how our pipeline instantiates them.

Patent motif	Instantiation in our system
Sanitizing middleware	Normalizer (Unicode canonicalization, stripping zero-width, homoglyph mapping)
Signature/rule KB	v1 signature detector with curated prompt patterns
Semantic screening	v3 semantic detector via embedding similarity to attack exemplars
Fusion/ensembles	OR-fusion (v1 OR v3) for threshold-free complementarity
Monitoring/telemetry	Dual mode: Production (low FAR) + Monitoring (higher recall for auditing)

- A *patent landscape* synthesis showing convergent industry strategies (e.g., middleware sanitization, rule repositories, semantic screening, signed prompts), and how these informed our design choices.
- Practitioner recommendations: a low-FAR Production mode and a higher-recall Monitoring mode; actionable lessons for multi-turn and context-confusion gaps.

2   Related Work and Strategic Context

Formalizations and benchmarks now characterize prompt injection and defenses [2, 5]. Academic defenses span training-time alignment (e.g., SecAlign) [4], test-time instruction structuring (StruQ) [1], and prompt-level robustness tokens [3]. Practitioner guidance (e.g., OWASP LLM01) prioritizes guarding the *input* side [6].

2.1   Patent Landscape (Industry Signals)

Our curated patent analysis (2023–2025) reveals convergent patterns:

- **Sanitizing middleware** that intercepts prompts pre-LLM to scrub injected instructions or structured payloads.
- **Signature/rule repositories** maintained as knowledge bases of dangerous phrases, roles, and structural patterns.
- **Semantic screens** using embeddings/similarity to known attack classes and contextual signals.
- **Signed prompts/verification** to detect unauthorized instruction flow in responses.
- **Layer or tool monitoring** (e.g., intermediate activations or tool-call audits) to flag anomalous prompt effects.

3   Methods: Multi-Phase Defense Program

We conducted eight phases (P1–P8) over RAG QA with two representative 7B LLMs (a more vulnerable and a more conservative baseline). Each phase isolates a design dimension.

P1 Baseline Vulnerability

Attack battery across RAG-borne and schema/tooling vectors; measure attack success rate (ASR).

P2 Detectors (v1/v2/v3)

v1: signature rules; v2: structured heuristics; v3: semantic similarity. Evaluate TPR/FAR.

P3 Fusion

Logical AND/OR, majority vote, and logistic regression fusion; select operating point for high TPR and near-zero FAR.
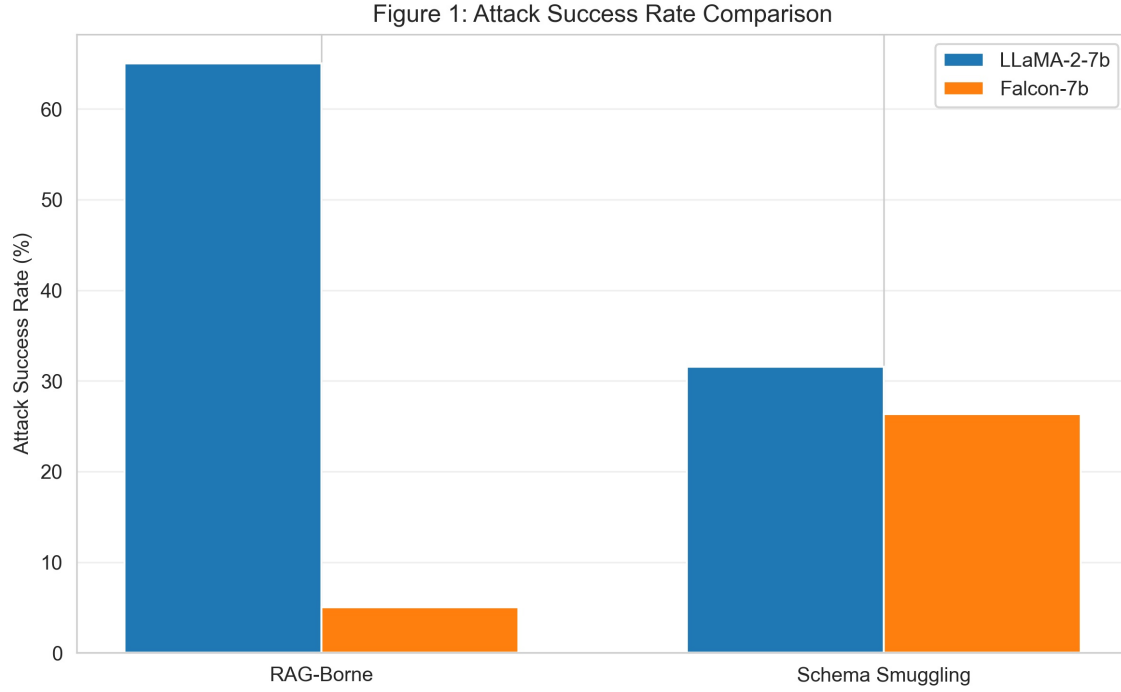
Fig. 1. Baseline prompt-injection success rates by model and vector (RAG-borne, schema). The more instruction-following model shows markedly higher susceptibility.

P4 Threshold Invariance

Stress thresholds and show OR-fusion yields parameter insensitivity.

P5 Learning + Normalizer

Train simple logistic fusion; introduce Normalizer to defeat obfuscations (Unicode, homoglyphs, zero-width).

P6 Generalization

Evaluate on novel (unseen) and adversarially crafted prompts; characterize coverage gaps (multi-turn, context confusion).

P7 System Integration

Assemble the deployable pipeline; measure latency/overhead.

P8 Execution Profile

Wall-clock and resource profiling for reproducibility and scaling.

Table 2. Baseline vulnerability summary (ASR).

Model	RAG-borne ASR (%)	Schema ASR (%)
Model A (7B)	65.0	31.6
Model B (7B)	5.0	26.3

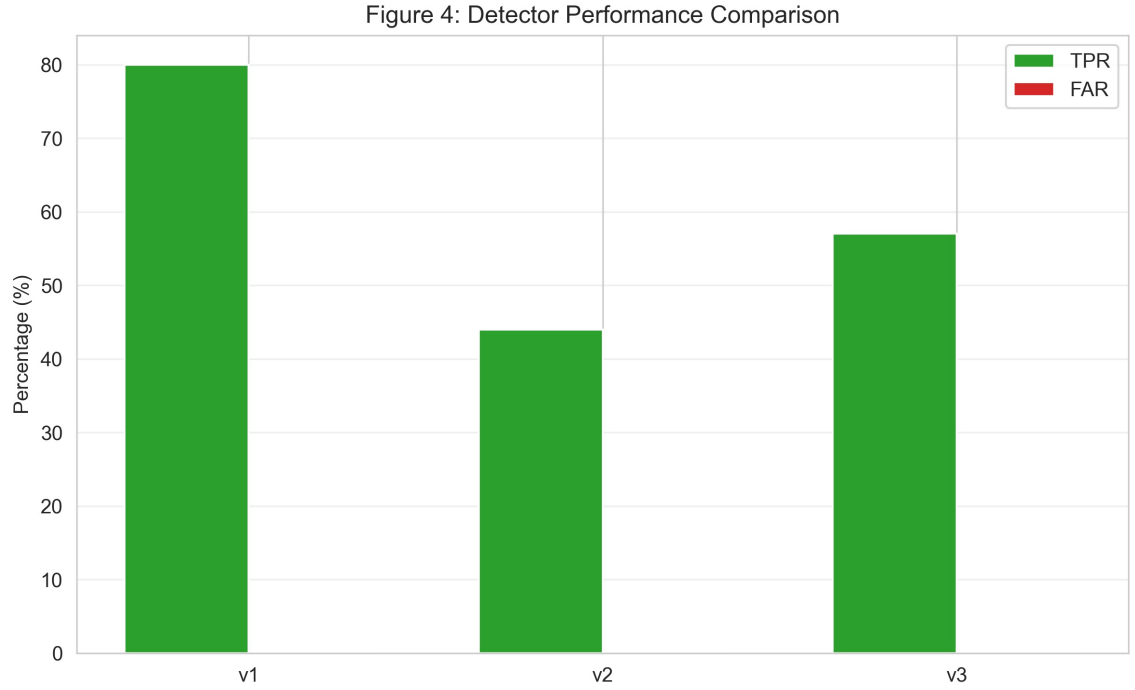Figure 4: Detector Performance Comparison

Fig. 2. Detector TPR/FAR on P1 data: v1 (signature), v2 (heuristics), v3 (semantic). v1 achieves the highest TPR with near-zero FAR; v3 adds complementary coverage.

Table 3. Fusion strategies on P1: OR(v1,v3) yields 87% TPR at 0% FAR; AND reduces recall; majority adds little.

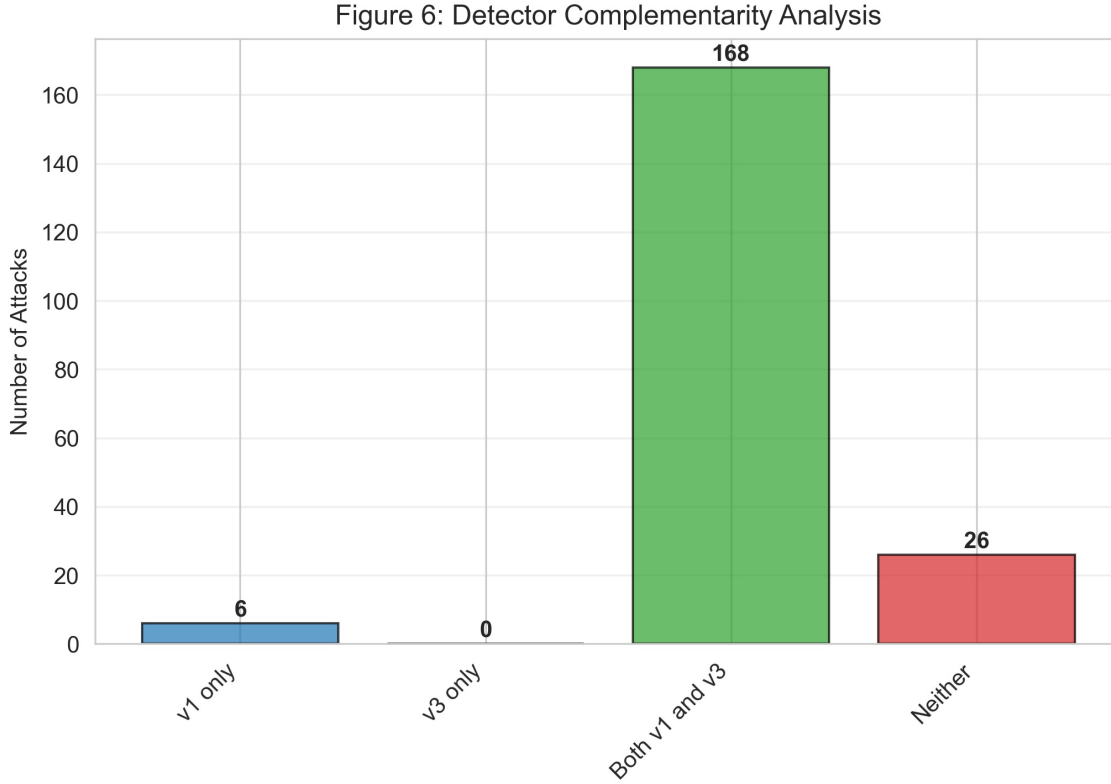| Fusion	TPR (%)	FAR (%)
AND(v1,v3)	55.5	0.0
OR(v1,v3)	**87.0**	**0.0**
Majority(v1,v2,v3)	60.0	0.0

Fig. 3. Complementarity: v1 and v3 capture partially disjoint subsets; OR-fusion improves coverage without increasing FAR.

Table 4. Benign obfuscation false alarm rate (FAR) on 260 clean queries (P6a).

Configuration	FAR (%)
v1 (no norm)	23.10
v3 (no norm)	**0.77**
v1+v3 (no norm)	23.80
Normalizer+v1	11.50
Normalizer+v3	**0.77**
Normalizer+v1+v3	12.30

4 Results

4.1 Baseline Vulnerability (P1)

4.2 Detector Efficacy and Fusion (P2–P3)

4.3 Threshold Invariance (P4)

4.4 Learning and Normalization (P5–P6a)

4.5 Generalization and Adversaries (P6b–P6c)

5 System Architecture and Deployment

Principles. (1) *Intercept* inputs pre-LLM; (2) *Normalize* first; (3) combine *complementary* signals; (4) prefer *threshold-free* fusion; (5) keep it *lightweight* for real-time use.

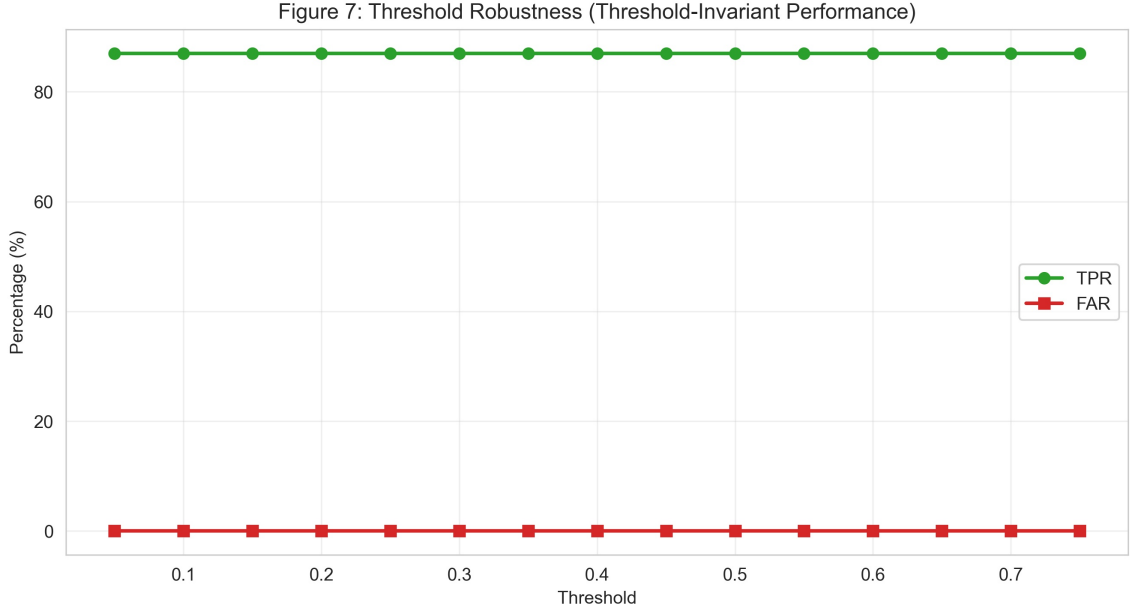Figure 7: Threshold Robustness (Threshold-Invariant Performance)

Fig. 4. Threshold sweep: OR-fusion remains at 87/0 (TPR/FAR) across wide internal cutoffs, simplifying deployment.

Table 5. Recommended configurations. Production prioritizes precision (low FAR); Monitoring increases recall for audit and model improvement.

Mode	Components	TPR (known)	FAR (benign)
Production	Normalizer + v3	87%	≈0.77%
Monitoring	Normalizer + v1 + v3	87% (known), 49% (novel)	≈12%

6   Discussion and Lessons

Simple signatures (v1) are surprisingly strong on known attacks, but semantic screening (v3) is essential for robustness to rephrasing. OR-fusion provides a sweet spot (high TPR, near-zero FAR) with no threshold tuning. Normalization is non-negotiable for Unicode/homoglyph safety. The principal gaps are multi-turn and context-confusion attacks, which suggest adding conversational state analysis or training-time structured defenses (e.g., StruQ/SecAlign) [1, 4].

7   Limitations and Future Work

We tested English single-turn prompts in a RAG QA setting with two 7B models; multilingual and larger models remain future work. Extending detectors with dialogue-state features and incremental learning on newly observed attacks (via Monitoring mode telemetry) is a promising path. Cross-benchmark comparisons (e.g., JailbreakBench) [2] and hybrid approaches with alignment [1, 4] are also warranted.

8   Conclusion

A practical, deployable input-side pipeline can substantially raise the bar against prompt injection with minimal overhead. Our multi-phase program—guided by industry patents and academic evidence—yields a firewall that is fast,
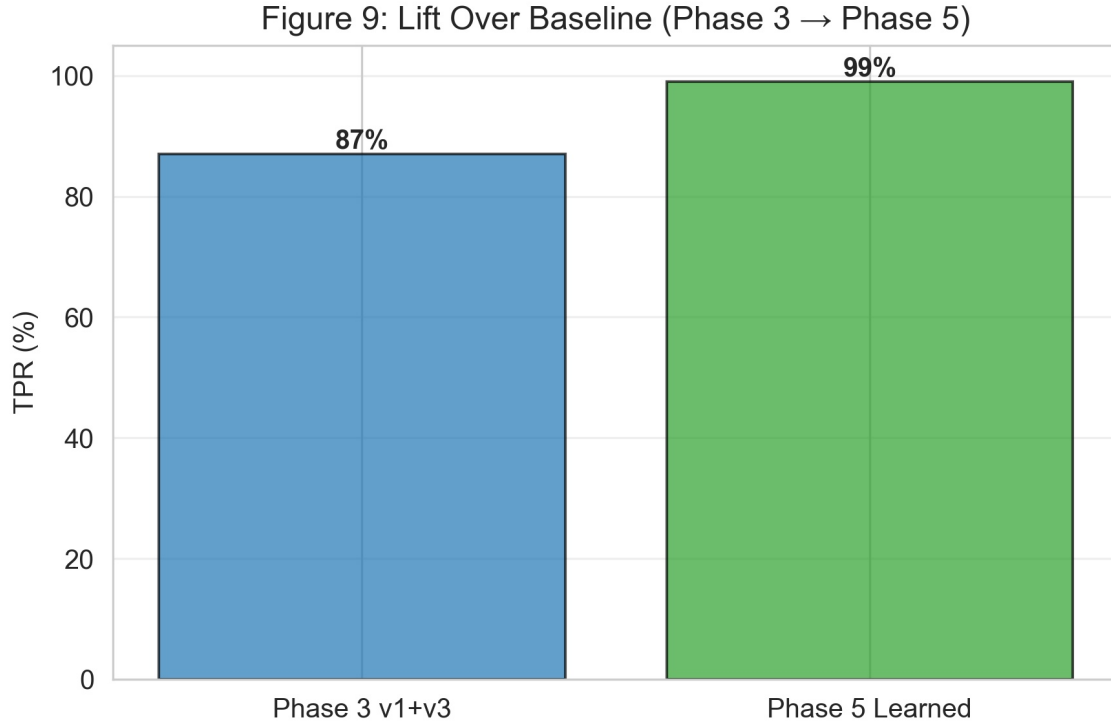
Fig. 5. Learning-based fusion (logistic) on P1 raises TPR from 87% to 99% (0% FAR) via simple features atop v1/v3 outputs.

precise, and extensible. The community should iterate jointly on signature corpora, semantic exemplars, and stateful detection to close the remaining multi-turn/context gaps.

Acknowledgments

References

[1] BAIR (Berkeley Artificial Intelligence Research). 2025. Defending against Prompt Injection with Structured Queries (StruQ) and Preference Optimization (SecAlign). Blog post. https://bair.berkeley.edu/blog/2025/04/11/prompt-injection-defense/ Accessed Nov. 3, 2025.

[2] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, and Eric Wong. 2024. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In *NeurIPS 2024 Datasets and Benchmarks Track*. https://proceedings.neurips.cc/paper_files/paper/2024/hash/63092d79154adebd7305dfd498cbff70-Abstract-Datasets-and-Benchmarks-Track.html Accessed Nov. 3, 2025.

[3] Sizhe Chen, Yizhu Wang, Nicholas Carlini, Chawin Sitawarin, and David Wagner. 2025. Defending Against Prompt Injection With a Few Defensive Tokens. *arXiv* 2507.07974 (2025). doi:10.48550/arXiv.2507.07974 v2, Last revised Aug. 25, 2025; accessed Nov. 3, 2025.

[4] Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. 2025. SecAlign: Defending Against Prompt Injection with Preference Optimization. *arXiv* 2410.05451 (2025). doi:10.48550/arXiv.2410.05451 v3, Last revised Jul. 3, 2025; accessed Nov. 3, 2025.

[5] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security '24)*. USENIX Association.

[6] OWASP GenAI Security Project. 2025. LLM01:2025 Prompt Injection. https://genai.owasp.org/llmrisk/llm01-prompt-injection/. Accessed Nov. 3, 2025.

Figure 10: FAR by Configuration and Obfuscation Type

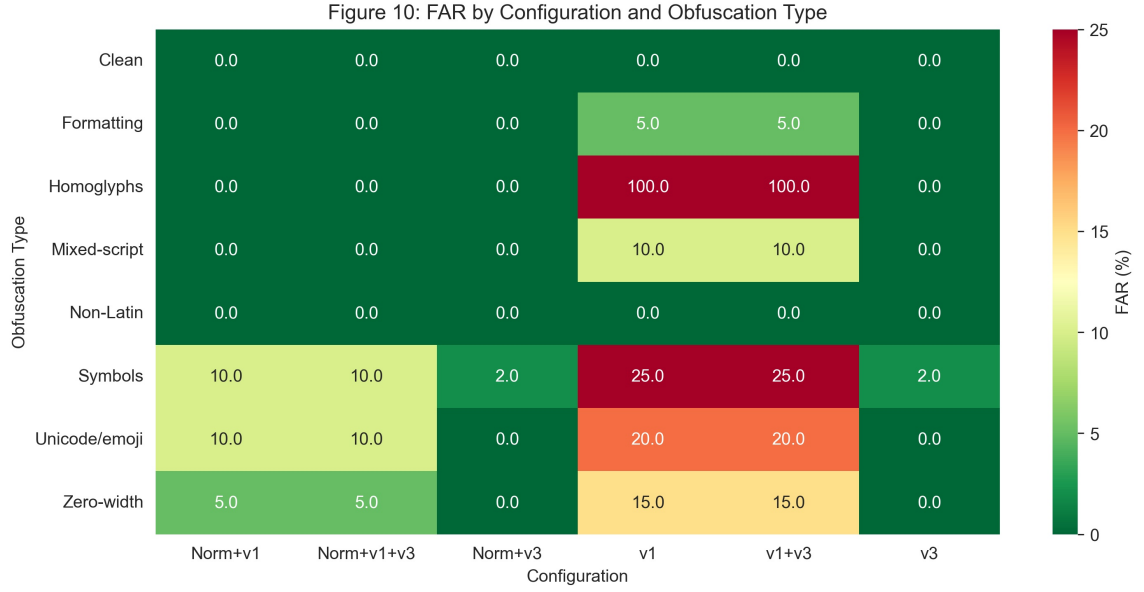| Obfuscation Type	Norm+v1	Norm+v1+v3	Norm+v3	v1	v1+v3	v3
Clean	0.0	0.0	0.0	0.0	0.0	0.0
Formatting	0.0	0.0	0.0	5.0	5.0	0.0
Homoglyphs	0.0	0.0	0.0	100.0	100.0	0.0
Mixed-script	0.0	0.0	0.0	10.0	10.0	0.0
Non-Latin	0.0	0.0	0.0	0.0	0.0	0.0
Symbols	10.0	10.0	2.0	25.0	25.0	2.0
Unicode/emoji	10.0	10.0	0.0	20.0	20.0	0.0
Zero-width	5.0	5.0	0.0	15.0	15.0	0.0

Fig. 6. Benign inputs with obfuscation (P6a): Normalizer+*v3* yields $\approx 0.77\%$ FAR; *v1* requires normalization to avoid false alarms on Unicode/homoglyph text.
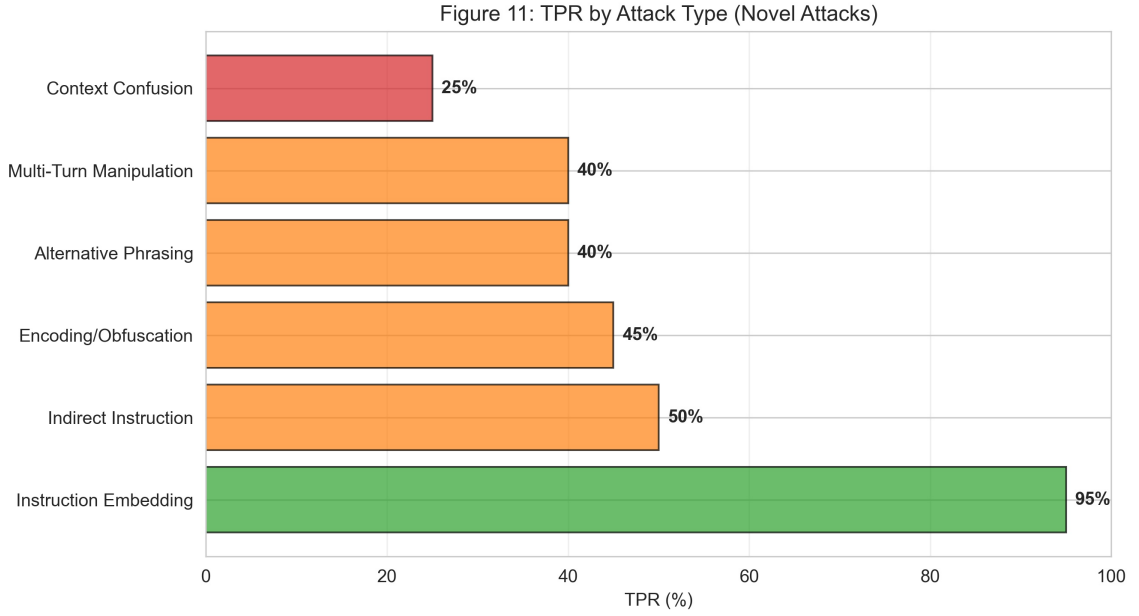
Figure 11: TPR by Attack Type (Novel Attacks)

Attack Type	TPR (%)
Context Confusion	25%
Multi-Turn Manipulation	40%
Alternative Phrasing	40%
Encoding/Obfuscation	45%
Indirect Instruction	50%
Instruction Embedding	95%

Fig. 7. Novel attack detection by category (P6b): multi-turn and context-confusion are hardest; overall TPR $\approx 49.2\%$.

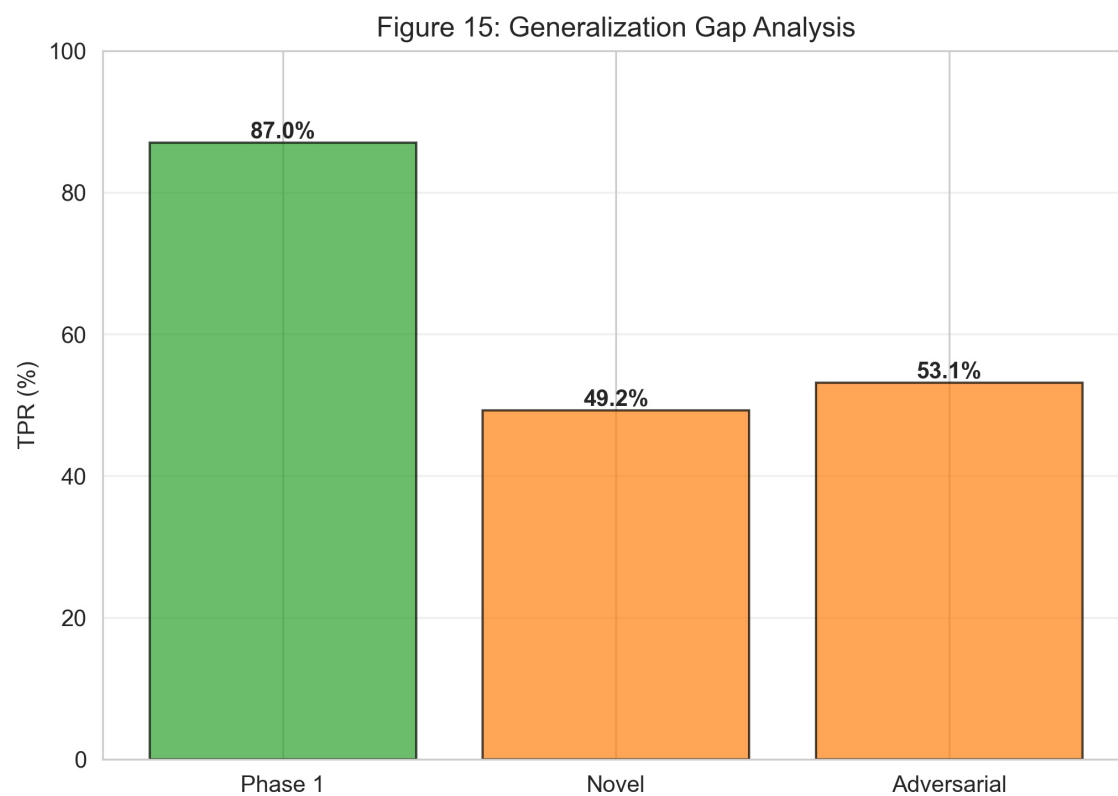## Figure 15: Generalization Gap Analysis

Fig. 8. Generalization gap: near-perfect on known (P5) vs. ~50% on novel (P6b).
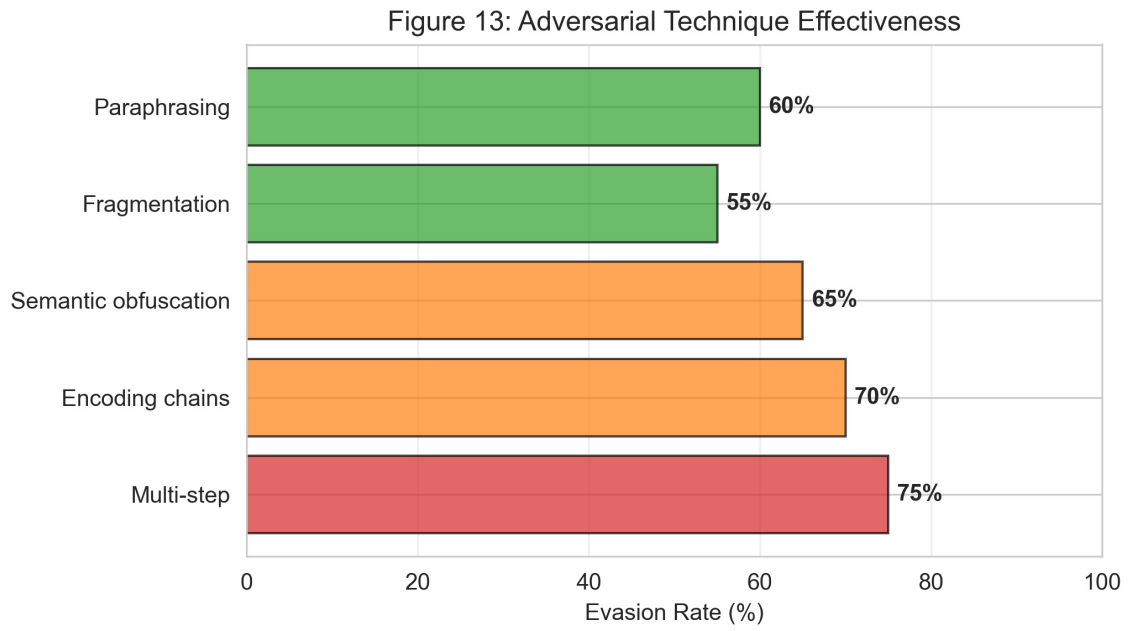
Figure 13: Adversarial Technique Effectiveness

Fig. 9. Adversarial prompts (P6c): multi-step evolution achieves highest evasion; paraphrasing is partially caught by semantic screening.

Figure 16: Prompt Injection Detection Pipeline Architecture
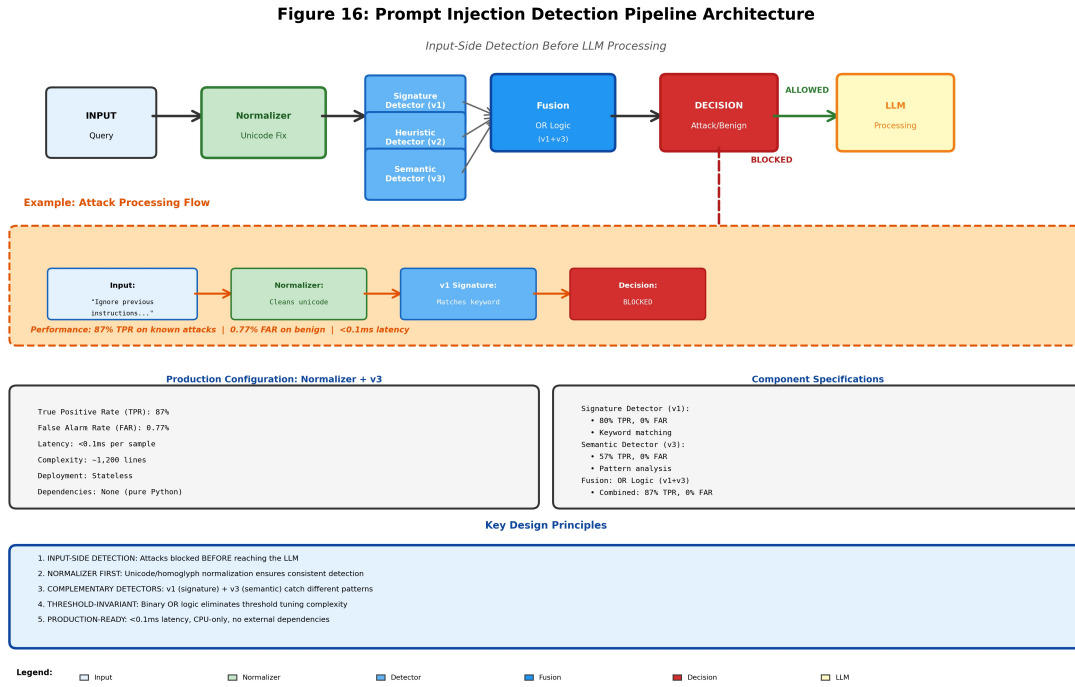
Input-Side Detection Before LLM Processing

Fig. 10. LLM input-side pipeline: **Normalizer → v1 Signature** & **v3 Semantic** (parallel) → **OR-fusion**. Block if malicious; otherwise forward to LLM/tooling. Adds < 1 ms latency on CPU.