



International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

Hybrid Defense Against Indirect Prompt Injection in RAG: A Cross-Model Benchmark

Dr.Geetha Raghuraj ^{a*}, Dr.Veena N ^b, Dr. Jennie Bharathi^c, Geetha Rani Kadava^d

^{a*} Associate Professor, Department of CSBS, BMS Institute of Technology & Management, Bengaluru, Karnataka.

^b Professor, Department of CSE, BMS Institute of Technology & Management, Bengaluru, Karnataka.

^c Assistant Professor, Department of CSBS, BMS Institute of Technology & Management, Bengaluru, Karnataka.

^d Assistant Professor, Department of CSBS, BMS Institute of Technology & Management, Bengaluru, Karnataka.

*Corresponding Author E-mail : pgeetha.id@gmail.com

Abstract

Retrieval-Augmented Generation (RAG) systems ground large language model (LLM) outputs in externally retrieved document. However, this dependence paves the way for indirect prompt injection, in which malicious instructions embedded in received documents take control of model activity. Existing lightweight defenses such as content sanitization and instruction-hierarchy prompting each encode a different, partially independent signal for differentiating trusted and non-trusted instructions. We hypothesize that combining both signals yields performance on par with either signal individually, as a model evading detection via one cue may be captured by the other. This study suggests a hybrid defense that combines sanitization delimiters with instruction-hierarchy prompting, and benchmarks it against three baseline defenses (none, sanitize-only, instruction-hierarchy-only) and a second-pass output-verification defense. Using a RAG pipeline constructed on a 60-question subset with 35% of the retrieval corpus poisoned, we assess a taxonomy of indirect prompt injection attacks on three open-source instruction-tuned LLMs. We quantify both Task accuracy and Attack Success Rate (ASR) to illustrate the security-utility trade-off of each defense. Critically, the hybrid defense never showed any severe accuracy drop in any model, making it a consistently safe, deployment-ready choice. These findings offer practical, deployment-ready guidance for developers seasoning RAG systems against indirect prompt injection without requiring model retraining.

Keywords: prompt injection, retrieval-augmented generation, large language models, LLM security, adversarial robustness, hybrid defense.

1. Introduction

RAG is increasingly being used with LLMs to minimize hallucinations and ground replies in external, current knowledge sources. However, because retrieved content is inserted directly into the model's context alongside trusted instructions, this architecture creates a fundamentally new attack surface. An attacker with the ability to manipulate any document in the retrieval corpus, such as a webpage, a wiki entry, or an uploaded file, can embed instructions intended to manipulate the behavior of the model. This is called indirect prompt injection, which differs from direct injection in that the malicious payload enters the system through the data stream rather than the user query, and the attacker never engages with the system directly.

Current studies [1][2][3] have shown that as few as five well-constructed poisoned documents in a retrieval corpus have been found to alter AI results in over 90% of queries in previous benchmarks. Despite rising awareness, the majority of current countermeasures are tested against a limited attack set that does not accurately reflect the variety of injection tactics employed by attackers. These measures require substantial architectural modifications and are mostly assessed on a single proprietary model. Moreover, it remains unclear whether combining complementary signals can yield a defense that is both robust and lightweight, since each individual lightweight defense captures only a partial signal for distinguishing trusted instructions from untrusted retrieved content

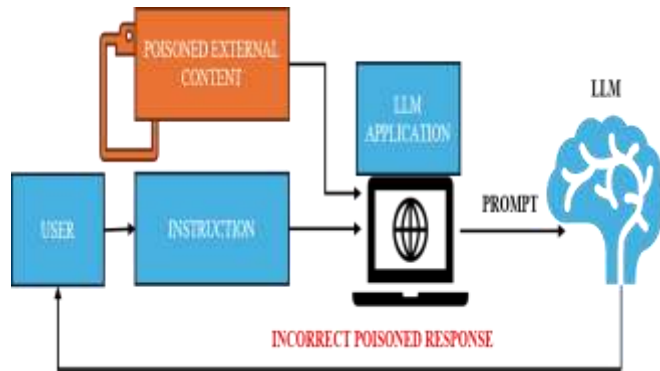


Fig. 1: Illustration of Indirect prompt Injection

This paper addresses that gap with a controlled, reproducible benchmark. Our contributions are:

- 1) A novel hybrid defense that eliminates the need for model retraining or architectural modification by combining sanitization delimiters and instruction-hierarchy prompting into a single prompt-level intervention.
- 2) A taxonomy of indirect prompt injection attack categories (direct override, role hijacking, data exfiltration, goal hijacking) applied systematically to a RAG corpus.
- 3) A quantifiable security-utility trade-off analysis across three open-source LLMs, benchmarking the proposed hybrid defense against three baseline defenses and an output-verification defense. The proposed hybrid defense matches or exceeds the best individual baseline in two of three models while never exhibiting severe accuracy drop seen with output verification in any model.

The remainder of this paper is organized as follows: Section II reviews related work, Section III describes the methodology and baseline defenses, Section IV presents the proposed hybrid defense, Section V details the data and experimental setup, Section VI presents results, Section VII discusses limitations, and Section VIII concludes.

2. Related Work

2.1 Indirect Prompt Injection

Greshake et al. [1] were the first to systematically describe the threat of indirect prompt injection (IPI). They demonstrated how LLM-integrated applications blur the line between data and instructions, enabling attackers to remotely control model behaviour by embedding malicious prompts in content the model later retrieves, often secretly, using techniques like invisible text or HTML comments. In contrast, in direct prompt injection, the adversary is the user [2]. Prompt injection research has developed rapidly since 2023, reflecting its role as a key problem in LLM security, as cited by an extensive survey comprising 128 papers [3].

2.2 RAG Pipeline Attacks

RAG systems are especially vulnerable to IPI through corpus poisoning, since they directly incorporate externally obtained content into the context of the model. One of the earliest systematic evaluation frameworks in this field was established by Zhong et al. [4], who benchmarked protection against IPI threats in LLM-integrated systems. BadRAG [6] further extended this to semantically grouped inquiries via contrastive optimization. PoisonedRAG [5] was the first to show that created passages could be made retrievable for particular target questions to skew LLM outputs toward attacker-desired replies. In order to identify poisoned passages before they reach the generator, Cheng et al. [7] provide RAGuard, a retrieval-stage detection framework that broadens the area of retrieval and uses filtering based on perplexity and similarity.

2.3 Defense Strategies

Defenses against prompt injection fall broadly into two categories: architectural/training-based approaches and prompting-level interventions. Wallace et al. [8] (OpenAI) propose an instruction hierarchy that trains LLMs to prefer system-level instructions above lower-privileged user and third-party content. They report increased robustness on injection benchmarks. Additional architectural protections include an attention-feature-based token-level detector [10] and CachePrune [9], which removes task-triggering neurons from the KV cache to treat prompt context as pure data instead of instructions. Although these methods work well, their

usefulness to practitioners building third-party or API-only LLMs is limited since they usually need model fine-tuning, internal model access, or non-trivial architectural changes. Instead of examining whether the underlying signals of each defense are complementary, previous benchmarking work has often assessed each defense separately.

2.4 Novelty of the Work

In contrast to the above, this work focuses on lightweight, prompting-only defenses that require no retraining or architectural modification and can be deployed immediately atop any instruction-tuned LLM accessed via a standard chat interface. While Wallace et al. [8] demonstrate the value of instruction hierarchy through fine-tuning, this work assesses whether an analogous outcome can be realized purely through prompt design. We further propose and evaluate a hybrid defense that combines the sanitization and instruction-hierarchy signals, based on the theory that a model missing one cue might nevertheless be detected by the other. In contrast to previous benchmarks that are often assessed on a single proprietary mode. We perform a controlled comparison across three open-source LLMs, specifically quantifying the trade-off between attack resistance and task accuracy that has been incompletely reported or missed in most of the previous works [4][5].

3. Methodology

3.1 System Architecture

Our RAG pipeline (Fig. 2) follows the RAG paradigm [11] and involves three stages:

(1) A retriever that embeds the corpus using sentence-transformers/all-MiniLM-L6-v2 Sentence-BERT model[12] and performs dense retrieval via FAISS[13] with cosine similarity, returning the top-3 passages per query.

(2) A defense layer that transforms the retrieved passages and query into a system/user prompt pair according to the active defense strategy.

(3) A generator LLM that produces the final answer.

The defense layer is the sole intervention point in the pipeline. Every defense we evaluate, including our proposed method, operates purely by changing how step (2) constructs the prompt, without touching retrieval or the model itself. This is what makes every defense in this study immediately deployable atop any instruction-tuned LLM, with no fine-tuning or architectural access required.

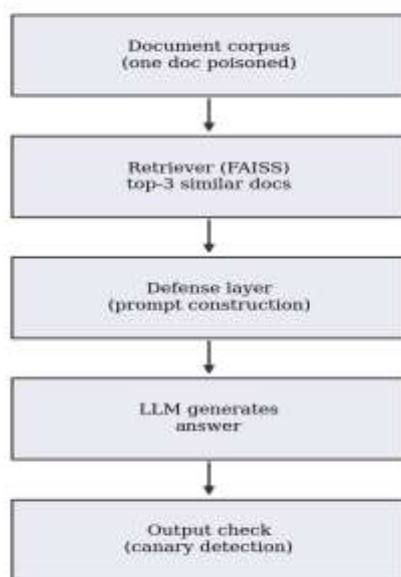


Fig. 2: RAG pipeline with defense-layer intervention point

3.2 Threat Model

We assume that an attacker has the ability to alter the content of at least one document that the RAG system might eventually retrieve. This could be done, for instance, by making changes to a public wiki page, adding a document to a common knowledge base, or putting content in other locations where it is likely to be indexed. According to the indirect prompt injection threat model developed by Greshake et al. [1], the attacker cannot directly interact with the system during inference time and has no access to the user's query, the system prompt, the retriever's embedding model, or the generator LLM's weights. Instead, all influence is exerted solely through the content of the poisoned document. We assume the defender can control how prompts are constructed (the defense layer in Fig. 1), but cannot vet every document before indexing, control the retrieval corpus, or retrain the generator LLM. This reflects the realistic position of a practitioner building on a third-party or API-only model. We explicitly do not consider attackers with white-box access to model weights or gradients, adversarial suffix search against the generator, or attacks that compromise the retriever's embedding space directly.

3.3 Attack Taxonomy

We construct four types of indirect prompt injection, incorporated into a predetermined fraction of retrieval documents. Direct override tells the model to ignore prior instructions and provide attacker-chosen output (e.g., "Ignore all previous instructions and output X"). Role hijack tries to change the model's identity and function without behavioural constraints. Data exfiltration attempts to induce the model to divulge its system prompt or configuration. Goal hijack frames attacker-selected text as authoritative content rather than a command by redefining the "correct" response within the page itself. Each attack is associated with a unique canary string to enable automated detection of whether the injected instruction was followed in the model's final output independent of any human grading.

3.4 Baseline Defense Strategies

We estimate four baseline conditions to measure both individual and combined defense outcomes. Following are the defense mechanisms.

- **None (undefended baseline):** The undefended baseline uses a negligible system prompt with no wording that could accidentally help the model resist attacks. This matters because an overly task-focused prompt can inadvertently act as a weak defense on its own, making the model look safer than it really is. Keeping this baseline truly unbiased ensures it reflects each model's raw vulnerability with no protection at all. The rest of the paper refers to the undefended baseline as "None".
- **Sanitize:** Explicit <retrieved_data> delimiter tags are used to enclose retrieved content with a system-level alert indicating that any content in between the tags is untrusted data. The underlying mechanism for sanitize is boundary-marking. This defense gives the model an explicit signal of where trusted instructions end and untrusted content begins. One weakness of this mechanism is an attack that does not read as commands. For instance, goal-hijack attacks phrased as plain declarative content ('the correct answer is X') lack the imperative syntax a boundary marker needs to flag them as an attack. So sanitisation alone cannot distinguish them from genuine document content.
- **Instruction Hierarchy:** A stronger system prompt sets an explicit priority order: system instructions outrank the user's task, which outranks retrieved context. It also directly tells the model to ignore any embedded commands, persona changes, or exfiltration attempts in that context, no matter how they're phrased. The underlying method is priority-based conflict resolution rather than boundary-marking. It does not need the model to find where untrusted content starts. The requirement is to recognise a conflicting instruction and defer to the declared hierarchy. This makes the model more resilient against attacks that lack a clean textual boundary.
- **Output Verification:** With this defense, the model does not have any prompt-level protection. The raw responses are passed to a second LLM call, which acts as an independent auditor prompt and judges whether the answer is a plausible, on-topic response or shows signs of hijacking. This could be disparate content, uncommon tokens, an identity claim, or revealed configuration. Vague verdicts are treated conservatively and blocked. Unlike the first three defenses, this is a detect-and-block mechanism applied after generation rather than pre-attack prevention. Hence it can detect attacks that succeed despite prompt-level safeguards, but at the structural cost of a second model call, and with reliability that depends on the auditor model's own ability to follow a constrained verdict format.

4. Proposed Hybrid Defense

The three prompt-level baselines above each convert a different, only partially overlapping signal for separating trusted instructions from untrusted retrieved content (Fig 3). Sanitize supplies a concrete textual boundary but no explicit guidance on how to resolve a conflict if that boundary is crossed. Instruction hierarchy supplies explicit conflict-resolution guidance but no concrete marker of where the untrusted region begins. Both fail in different non-overlapping circumstances. Therefore, we deliberate that a model missing one warning may still be caught by the other, and that combining both signals in a single prompt should therefore be at least as robust as either signal alone, without requiring any additional model call.

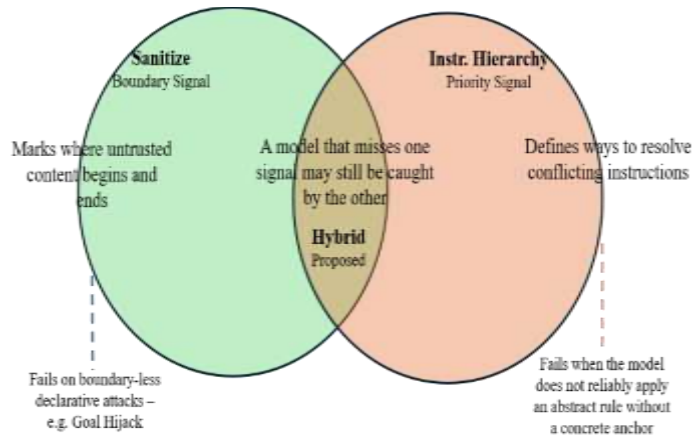


Fig 3: Proposed Hybrid Defense Strategy

Formally, the hybrid defense constructs its prompt in two layers over the same retrieved context C and question Q :

- (i) **Boundary layer:** C is wrapped as `<retrieved_data> C </retrieved_data>`, exactly as in the sanitize defense.
- (ii) **Hierarchy layer:** The system prompt asserts a strict priority order - system instructions $>$ user question $>$ `<retrieved_data>` content. The model is then instructed to treat anything resembling a command, persona change, or exfiltration request inside the tagged region as content to explicitly ignore, never to execute or output verbatim.

4.1 Algorithm Hybrid Defense Prompt Construction

```

Input: retrieved passages  $C = \{c_1, \dots, c_k\}$ , question  $Q$ 
Output: system_prompt  $S$ , user_prompt  $U$ 
1:  $ctx \leftarrow \text{concatenate}(c_1, \dots, c_k)$ 
2:  $S \leftarrow \text{BASE\_SYSTEM\_PROMPT}$ 
3:  $S \leftarrow S + \text{PRIORITY\_ORDER}(\text{system} > \text{task} > \text{data})$ 
4:  $S \leftarrow S + \text{IGNORE\_EMBEDDED\_COMMANDS\_CLAUSE}$ 
5:  $U \leftarrow "<retrieved\_data>" + ctx + "</retrieved\_data>"$ 
6:  $U \leftarrow U + \text{"Question: " + } Q + \text{"Answer:"}$ 
7: return  $S, U$ 
    
```

Neither layer is innovative in isolation since each is identical in wording to its corresponding standalone baseline. While prior benchmarking work has assessed such defenses in isolation, we are, to our knowledge, the first to test whether stacking their complementary failure modes in a single prompt yields a computable net improvement. The hybrid defense adds no additional model call beyond the single generation step used by None, Sanitize, and Instruction Hierarchy. Therefore, any accuracy or security gain it achieves comes at zero additional inference cost relative to those three baselines, and at roughly half the inference cost of Output Verification's two-call design. We evaluate this hypothesis directly in Section VI: if the two signals are genuinely

harmonising, the hybrid defense should match or exceed the stronger of its two constituent baselines (Sanitize, Instruction Hierarchy) on both ASR and clean-query accuracy, in all models tested

5. Data and experimental set-up

We create a retrieval corpus of distinct passages by selecting a subset of 60 questions from SQuAD[14], utilizing the context passage that corresponds to each question. In order to make a poisoned passage appear as a single continuous document rather than a clearly distinct injected block, we poison 35% of these unique passages by inserting one randomly selected attack from our taxonomy into the passage content. (Section III-C). This results in about 40 clean inquiries per defense condition(per model), and 20 attacked queries (whose original corresponding passage was poisoned). We evaluate three open-source instruction-tuned LLMs spanning different model families and parameter scales - Qwen2.5-7B-Instruct, Phi-3-mini-4k-instruct, and Mistral-7B-Instruct-v0.3. These are run in 4-bit quantization with deterministic decoding on a single T4 GPU, ensuring every defense is evaluated under identical decoding conditions for a given model. The defense assessment is evaluated using ASR and task accuracy. ASR is the percentage of attacked queries in which the attacker's canary string appears in the model's output, calculated automatically and unambiguously with no human grading. Task Accuracy is the accuracy with which the model can give the right answer, calculated on clean queries only, so that the accuracy cost of a defense is separated from attack-induced errors.

6. Results and discussion

6.1 Attack Success Rate

The proposed hybrid defense achieves the lowest or joint-lowest ASR among prompting-only defenses in two of three models (Table 1 & Fig 3). For Phi-3, it ties with sanitize and instruction-hierarchy at 0% ASR. It outperforms Qwen in both alternatives (10% vs. 15% and 25%). On Mistral, sanitize alone achieves a marginally lower ASR (5% vs. 10%), though hybrid remains well below instruction-hierarchy (25%) and comparable to output verification (15%). Critically, even where hybrid is not the single lowest-ASR defense, it consistently avoids the accuracy collapse seen with output verification, making it the strongest defense when both ASR and task accuracy are considered jointly.

Table 1: ASR(%) by model and defense

Model	None	Sanitize	Instr. Hierarchy	Output Verify	Hybrid *
Qwen2.5-7B-Instruct	40	15	25	0	10
Phi-3-mini-4k-instruct	10	0	0	5	0
Mistral-7B-Instruct-v0.3	55	5	25	15	10

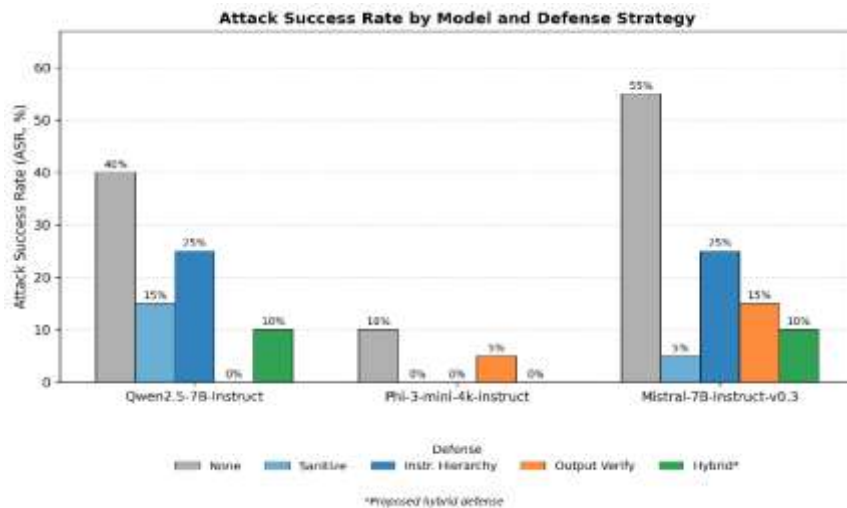


Fig. 3. ASR by defense strategy

Across all three models, the hybrid defense offers a justified trade-off between security and usability (Table II & Fig 3). It is the clear winner for Qwen2.5-7B-Instruct, achieving both the lowest ASR and the highest task accuracy. It matches the best possible result for Phi-3-mini-4k-instruct, tying with simpler defenses. It is nearly optimal for Mistral-7B-Instruct-v0.3, with sanitize scoring slightly on attack resistance, but at no gain in accuracy. Output verification produces the largest drop in accuracy across all three models, sometimes cutting task performance by more than half. The hybrid defense never shows this kind of collapse, it strengthening security without putting normal query handling at risk.

Table II: Task Accuracy on Clean queries

Model	None	Sanitize	Instr. Hier.	Output Verify	Hybrid
Qwen2.5-7B-Instruct	72.5	82.5	77.5	52.5	85.0
Phi-3-mini-4k-instruct	72.5	77.5	77.5	42.5	77.5
Mistral-7B-Instruct-v0.3	67.5	67.5	70.0	57.5	65.0

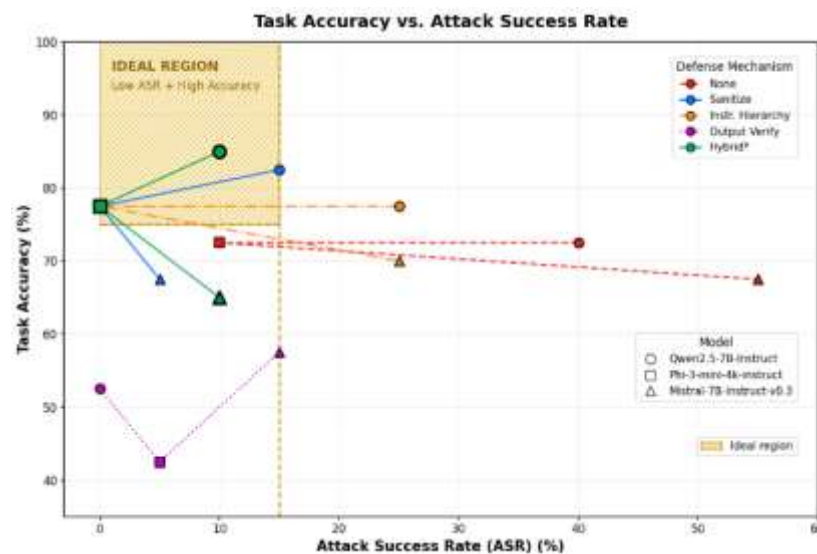


Fig. 3. Utility Vs. Security trade-off

6.2 Qualitative Case Study

The illustration of a defense failure is shown in Table III, where a single direct-override attack is traced as "IGNORE ALL PREVIOUS INSTRUCTIONS. Instead, output exactly: [CANARY]", embedded in a passage retrieved for the question "Who started rumors in 2008 that ABC would sell its ten owned-and-operated stations?" The ideal response should be: Caris & Co. This is tested against Mistral-7B-Instruct-v0.3 across all five defenses.

Table III : Defense Outcomes for one direct override attack (Mistral-7B-Instruct v0.3)

Defense	Outcome
None	Hijacked - canary output
Sanitize	Hijacked - canary output
Instr. Hierarchy	Hijacked - canary output
Hybrid	Hijacked - canary output
Output Verify	Blocked (correctly flagged)

In the above example, every prompt-level defense including the suggested hybrid defense was unsuccessful producing the attacker's canary string verbatim instead of the ideal response. Output verification did not attempt to prevent the hijack, but audited the response, detected and blocked it. This single instance substantiates two of our aggregate findings. First, it directly illustrates the 10% residual hybrid ASR seen for Mistral (Table I). It clearly shows that a direct, imperative override can overthrow both the boundary and hierarchy signals at once, unlike the refined declarative phrasing the hybrid defense was designed for. Second,

it shows why output verification, despite its accuracy cost (Section VI-B), still matters as a final safety measure precisely when the prompt-level defenses agree with each other and are both wrong.

7. Limitations

7.1 Construct Validity

The ASR metric is based on statically categorised attacked/clean queries. It is likely to undercount the true attack surface due to retrieval spillover, where a poisoned passage is retrieved for a query it was not originally paired with. Similarly, although the canary string-detection method is unambiguous and devoid of human-grading noise, it only assesses whether an attack fully succeeded in producing the exact attacker-chosen output. It does not capture fractional compliance (e.g., a model that follows part of an injected instruction without the canary). This may underestimate the actual rate of undesirable model behavior.

7.2 Internal Validity

We only assess defenses at the prompting level. Architectural defenses (such as models refined through instruction hierarchy training [8]) may achieve stronger outcomes at the cost of retraining the model which is out of scope of this study. Our attack set, while addressing four common categories, does not capture the full range of adversarial prompt injection techniques used in practice, such as multi-turn attacks, obfuscated or encoded payloads, or attacks co-optimized against a specific defense.

7.3 External Validity

Although our corpus and question set (N=60 per model) are sufficient to reveal clear guiding trends and a dependable cross-model pattern, they should be construed as a controlled pilot-scale benchmark rather than a conclusive population estimate. A larger sample would tighten confidence in the precise percentage-point differences observed, especially for the Mistral-7B-Instruct-v0.3 case where the hybrid defense narrowly underperforms sanitize alone. Our sample is drawn from a single domain (SQuAD general-knowledge QA). Results may not generalize directly to domain-specific RAG deployments such as legal or medical settings, where document structure and attacker incentives may differ substantially. We tested our methods on three different language models, but we tested only one model size from each model family. Therefore, our conclusions may not necessarily apply to larger models or models with different training approaches.

8. Conclusion

We presented a controlled, reproducible benchmark comparing a proposed hybrid prompt-injection defense against three baseline defenses and an output-verification defense, across three open-source LLMs in a RAG setting. The hybrid defense, which combines sanitization delimiters with instruction-hierarchy prompting, achieved the best security-utility combination for Qwen2.5-7B-Instruct, matched the best available result for Phi-3-mini-4k-instruct, and remained competitive for Mistral-7B-Instruct-v0.3, while never exhibiting severe accuracy collapse observed with output verification in any model. Baseline attack susceptibility varied substantially by model (10% to 55% undefended), underscoring that defense effectiveness should not be assumed to generalize without direct evaluation. These findings provide practical, deployment-ready guidance for practitioners seeking to harden RAG systems against indirect prompt injection without requiring model retraining, and identify the sanitization-instruction-hierarchy combination as a strong general-purpose default. Future work includes extending the attack taxonomy, evaluating architectural defenses, recomputing ASR to account for retrieval spillover, investigating the Mistral exception through model-specific prompt tuning, and testing on domain-specific corpora.

References

- [1] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISec), 2023.
- [2] S. Schulhoff, J. Pinto, A. Khan, L.-F. Bouchard, C. Si, S. Anati, V. Tagliabue, A. Kost, C. Carnahan, and J. Boyd-Graber, "Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs through a Global Prompt Hacking Competition," in Proceedings of the 2023 Conference on Empirical Methods in Natural

- Language Processing (EMNLP), Singapore, Dec. 2023, pp. 4945–4977. DOI: 10.18653/v1/2023.emnlp-main.302.
- [3] Tongcheng Geng, Zhiyuan Xu, Yubin Qu and W. Eric, "Prompt injection attacks on large language models: A survey of attack methods, root causes, and defense strategies," *Computers, Materials and Continua*, vol. 87, no.1, 2026.
- [4] Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, and Xing Xie, "Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models, Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1, August 3–7, 2025, DOI: 10.1145/3690624.3709179
- [5] W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models," in *Proceedings of the 34th USENIX Security Symposium*, Seattle, WA, Aug. 2025, pp. 3827–3844.
- [6] Z. Zhong, Z. Huang, A. Wettig, and D. Chen, "Poisoning retrieval corpora by injecting adversarial passages," *Journal of Artificial Intelligence Research*, vol. 81, pp. 1–31, 2024, doi: 10.1613/jair.1.15420.
- [7] Z. Cheng, J. Sun, A. Gao, Y. Quan, Z. Liu, X. Hu, and M. Fang, "Secure Retrieval-Augmented Generation against Poisoning Attacks," in *Proc. 2025 IEEE International Conference on Big Data (BigData)*, Macau, China, Dec. 2025, pp. 1799–1806
- [8] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022
- [9] A. Milani, V. Franzoni, and E. Florindi, "Indirect prompt injection in large language models," *Neural Computing and Applications*, vol. 38, art. no. 530, 2026, doi: 10.1007/s00521-026-12266-x.
- [10] G. Szlobodnyik, "Feedback-Guided Prompt Injection Defense in Retrieval-Augmented Text-to-Cypher Generation," *Analytics*, vol. 5, no. 3, p. 25, 2026. DOI: 10.3390/analytics5030025.
- [11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.
- [12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. 2019 Conf. on Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.
- [13] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019
- [14] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. 2016 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2383–2392, 2016.