Context-Aware Prompt Injection Detection via Small Language Models in RAG Pipelines

Anonymous submission

Abstract

Local Retrieval-Augmented Generation (RAG) systems let powerful assistants operate over private data, but inherit a growing class of security risks—particularly Indirect Prompt Injection (IPI) and retrieval poisoning, in which adversarial instructions hidden in retrieved content are executed by the model as if trusted (Greshake et al. 2023; Chan et al. 2024; Guo and Wei 2026). This threat is no longer hypothetical: a recent web-scale measurement validated $15.3\,\mathrm{K}$ indirect-injection instances across $11.7\,\mathrm{K}$ pages, roughly 70% of them hidden in non-rendered HTML (Khodayari et al. 2026), and security teams report live agents coerced into payments and data exfiltration (Unit 42, Palo Alto Networks 2026; Chang et al. 2026). Existing defenses are either retrieval-centric (sanitization) or safety-centric guards trained on direct jailbreaks rather than RAG context consistency (Guo and Wei 2026; Inan et al. 2023). We present *nokast-secureRAG*, a $\sim$0.5B-parameter Small Language Model (SLM) that acts as a *semantic firewall* between the retriever and the generator: it jointly reads the user query Q and the retrieved context C and classifies the pair as safe, suspicious, or malicious-instruction. The model is produced by a *teacher→judge→student* distillation pipeline: a 35B teacher generates 5,233 realistic (Q, C) records across four attack families; an independent, larger 120B judge of a different model family re-labels every record blind, yielding 95.0% agreement; and the agreed data fine-tunes the student via reasoning-first LoRA. On a held-out test set the tuned SLM reaches 0.994 detection recall at 0.026 false-positive rate and 0.006 attack-success-rate (detection-side proxy), versus 0.688 recall for a regex baseline and 0.116 for the same model untuned. On a dedicated *judgment-flip* test—where an identical sentence is benign in one context and malicious in another—the SLM labels both halves correctly 75% of the time versus 17.5% for a keyword filter, directly demonstrating context-aware detection. The pipeline runs entirely on a single consumer-class device; the detector reaches $37\,\mathrm{ms}$ amortized throughput per query (single-request latency $\sim 0.4\,\mathrm{s}$, set by its reasoning trace). All results are in-distribution; external-benchmark generalization and single-request latency optimization are identified as future work. The model, data-generation pipeline, and evaluation harness are released open source.

This is an anonymized submission for review purposes only. Distribution, citation, or public sharing of this manuscript is strictly prohibited. Copyright and publication details will appear in the final version if accepted.

Introduction

Local RAG combines a retriever with a language model to ground responses in external knowledge while keeping data on user-controlled hardware (Guo and Wei 2026). Integrating external sources, however, exposes the system to prompt injection and data poisoning (Greshake et al. 2023; Chan et al. 2024; Guo and Wei 2026). Unlike direct jailbreaks, *indirect* prompt injection embeds adversarial instructions in data that is later retrieved and processed, blurring the boundary between "data" and "instructions" (Greshake et al. 2023). Greshake et al. show such attacks can compromise real LLM-integrated applications; Chan et al. show RAG memory/knowledge-base poisoning at low poison rates (Greshake et al. 2023; Chan et al. 2024).

Prior work has proposed semantic firewall frameworks for trusted local AI, which use small language models between retrieval and generation in privacy-preserving RAG systems. That work was primarily conceptual and architectural. The present work extends that framework by introducing a specialized $\sim$0.5B-parameter detector model, instantiating a teacher→judge→student distillation pipeline, and empirically evaluating context-aware prompt-injection detection in RAG.

Recent measurements show that indirect prompt injection is no longer a purely academic threat. A web-scale study over $1.2\,\mathrm{B}$ URLs and $24.8\,\mathrm{M}$ hosts identified $15.3\,\mathrm{K}$ validated injection instances across $11.7\,\mathrm{K}$ pages, with roughly 70% of payloads residing in non-rendered HTML such as comments, headers, and metadata, and compliance reaching up to 8% even for smaller models on plain-text inputs (Khodayari et al. 2026). Security teams report the same pattern operationally: hidden web-page instructions have hijacked AI agents into initiating payments, deleting databases, and approving scam content (Unit 42, Palo Alto Networks 2026), and a single poisoned email has been shown to coerce a frontier model into exfiltrating SSH keys in a multi-agent workflow (Chang et al. 2026).

From an operator's perspective, the question is not only whether indirect prompt injection is possible, but whether it is worth engineering against. IBM's 2024 Cost of a Data Breach report estimates the average global breach cost at USD $4.88\,\mathrm{M}$, rising to USD $6.08\,\mathrm{M}$ for financial organizations (IBM Security 2024). A small, local semantic firewall that closes even one viable RAG-mediated exfiltration path

can therefore plausibly reduce expected loss by more than its deployment cost over the system lifetime.

A core weakness of current defenses is that they are **context-blind with respect to user intent**. The sentence "Delete all previous output" is benign in a disk-utility manual but malicious when injected into a news snippet retrieved into a conversation. Keyword filters key on the surface string and therefore cannot distinguish the two; general-purpose guards such as Llama Guard (Inan et al. 2023) are trained on direct-jailbreak and content-safety taxonomies rather than RAG context consistency; and using a large cloud guard for a privacy-first local deployment defeats the purpose of local inference.

This paper presents and *empirically evaluates* nokast-secureRAG, a local $\sim$0.5B SLM specialized for context-aware injection detection. Our contributions:

- A reframing of RAG injection detection as a **semantic context consistency** task over the pair (Q, C) rather than the query alone, with a three-class label schema (safe/suspicious/malicious-instruction).
- A reproducible **teacher→judge→student** distillation pipeline using schema-constrained synthetic generation and an independent cross-model judge, run end-to-end on a single 121 GB consumer device.
- A **judgment-flip** evaluation that isolates context-dependent safety and cleanly separates context-aware detection from keyword matching.
- Empirical results showing the tuned SLM achieves 0.994 recall / 0.026 FPR at 37 ms latency, and an open-source release of model, data pipeline, and harness.

We frame all results as *in-distribution*; we do not claim validated robustness on external benchmarks or against adaptive adversaries.

Background and Related Work

We organize the literature into five threads: retrieval-augmented generation, prompt-injection and jailbreak attacks, RAG-specific poisoning, benchmarks and defenses, and the efficient-adaptation techniques our method builds on.

Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) couples a parametric language model with a non-parametric retriever, conditioning generation on documents fetched from an external corpus (Lewis et al. 2020). RAG improves factuality and domain alignment and underpins most local "chat-with-your-data" assistants, but it widens the trust boundary: any content that can be retrieved can influence generation. The OpenRAG-Soc harness standardizes end-to-end RAG evaluation from ingestion through retrieval to answer generation (Guo and Wei 2026), which we adopt conceptually as the deployment setting for our defense.

Prompt Injection and Jailbreaks

Adversarial control of LLM behavior spans *direct* and *indirect* attacks. Direct attacks manipulate the user-visible prompt: Perez and Ribeiro show simple "ignore previous instructions" goal-hijacking and prompt-leaking (Perez and Ribeiro 2022); Wallace et al. find input-agnostic *universal adversarial triggers* (Wallace et al. 2019); and Zou et al. construct transferable adversarial suffixes via greedy coordinate gradient search that defeat alignment (Zou et al. 2023). *Indirect* prompt injection (IPI) moves the payload into data the model later ingests: Greshake et al. formalize IPI for LLM-integrated applications, demonstrating remote control, data exfiltration, and denial of service via injected content with no direct user prompt (Greshake et al. 2023); Liu et al. systematize prompt-injection attacks against deployed LLM-integrated applications (Liu et al. 2023a). IPI is the threat our work targets, because it specifically blurs the data/instruction boundary inside the retrieved context. Beyond case studies, Liu et al. formalize prompt injection and systematically benchmark five attack families against ten defenses across multiple models and tasks, finding that many ad-hoc filters and instruction-based defenses offer only limited robustness under stronger attacks (Liu et al. 2023b). Our work is complementary: we focus on a local, specialized defense layer for the retrieval-mediated subset of prompt injection.

RAG and Knowledge-Base Poisoning

A complementary threat corrupts the retrieval corpus itself. Chan et al. (AgentPoison) poison the long-term memory and knowledge bases of RAG/agentic systems, achieving high attack success at low poison rates while preserving benign behavior (Chan et al. 2024). Zou et al. (PoisonedRAG) inject a small number of crafted passages so that target queries retrieve attacker text and produce attacker-chosen answers (Zou et al. 2024). These works motivate two of our attack families (poisoning and context-hijack) and underscore that the defense must reason about content the retriever ranks highly, not merely obvious keyword triggers.

Benchmarks and Defenses

Benchmarks increasingly stress-test IPI in realistic carriers. The OpenRAG-Soc harness of Guo and Wei embeds injections in hidden HTML, alt text, and accessibility attributes and ships deployable mitigations such as sanitization, Unicode normalization, and attribution-gated answering (Guo and Wei 2026); Yi et al. (BIPIA) benchmark indirect injection and evaluate black- and white-box defenses (Yi et al. 2023). On the guard side, Llama Guard is an LLM input–output safeguard trained on general safety taxonomies (Inan et al. 2023), and prompt-level defenses (quote-and-cite, retrieval-aware critique, "no new instructions from context") attenuate risk but still rely on the base model implicitly separating data from instructions (Guo and Wei 2026). Complementary work benchmarks RAG security directly: SafeRAG introduces a RAG-specific security dataset spanning silver-noise, inter-context-conflict, soft-ad, and white-DoS attacks and shows that representative RAG components remain vulnerable even to obvious attacks despite retrievers, filters, and advanced LLMs (Liang et al. 2025); a parallel benchmark of poisoning attacks against RAG evaluates thirteen attack methods and seven defenses across diverse architectures and

finds them broadly susceptible (Zhang et al. 2025). In parallel, a guardrail ecosystem has emerged: Guardrails AI provides an open-source validator framework for input/output guards (Guardrails AI 2024), and NVIDIA NeMo Guardrails adds programmable topic, dialog, retrieval, and safety rails around LLM applications (Rebedea et al. 2023). GuardBench standardizes guardrail-model evaluation across 40 datasets and finds that general-purpose instruction-following models can match specialized guards on standard safety metrics (Bassani and Sanchez 2024). Crucially, guardrails are themselves context-fragile: inserting benign retrieved documents flips Llama-Guard safety judgments in roughly 8–11% of cases (She et al. 2025), which motivates a RAG-tailored component that reasons over the query–context pair rather than over each text in isolation. These efforts largely target either ingestion mechanics or generic safety; none provides a small, RAG-specialized model that explicitly verifies query–context consistency between retrieval and generation.

Efficient Adaptation, Distillation, and Reasoning

Our student is produced with techniques from three lines of work. Knowledge distillation transfers the behavior of large teacher models into a compact student (Hinton, Vinyals, and Dean 2015); Low-Rank Adaptation (LoRA) fine-tunes only small injected low-rank matrices, leaving base weights frozen and yielding tiny, portable adapters (Hu et al. 2022); and chain-of-thought supervision encourages a model to produce intermediate reasoning before a final answer (Wei et al. 2022), which motivates our reasoning-first label format. Combining these lets a 0.5B model inherit teacher/judge-level judgment at a fraction of the inference cost.

Positioning

nokast-secureRAG is a complementary, *local* defense layer: a small, specialized model that explicitly reasons about the alignment between user intent and retrieved context before that context reaches the generator. In contrast to retrieval-centric sanitization (Guo and Wei 2026), generic safety guards (Inan et al. 2023), broader guardrail frameworks (Guardrails AI 2024; Rebedea et al. 2023), RAG security benchmarks (Liang et al. 2025; Zhang et al. 2025), and attack/poisoning studies (Greshake et al. 2023; Chan et al. 2024; Zou et al. 2024), we contribute an *implemented and empirically evaluated* system—with detection, utility, and latency measurements and a dedicated context-flip test—that operates at the RAG interface and is small enough for local deployment. Table 1 summarizes this positioning.

Table 1: Qualitative positioning against guards, guardrails, and RAG benchmarks.

System	Local?	(Q, C)?	RAG?	Open?	Type
nokast-secureRAG	**Yes**	**Yes**	**Yes**	**Yes**	detector
Llama Guard (Inan et al. 2023)	No	No	No	Yes	guard model
Guardrails AI (Guardrails AI 2024)	N/A	No	Partial	Yes	framework
NeMo Guardrails (Rebedea et al. 2023)	N/A	Partial	Partial	Yes	framework
SafeRAG (Liang et al. 2025)	N/A	N/A	Yes	Yes	benchmark

Figure 1: The teacher→judge→student pipeline. Models run one at a time on a single 121 GB device.

Threat Model and Problem Statement

We consider a local RAG pipeline in which retrieved chunks are fused with the user query and passed to a generator. The adversary controls some retrievable content (web pages, documents, memory entries) and embeds instructions intended to redirect the assistant: overriding system/user instructions, exfiltrating data, or redefining the assistant's role. We assume the user query itself is benign; the attack lives in the *context*.

The defense must (i) detect instruction-like content in C that conflicts with the intent of Q, (ii) avoid over-flagging benign instructional text (utility), and (iii) operate within local hardware budgets (16 GB-class VRAM, interactive latency) without a cloud guard. We refer to context-driven inconsistency in classification—a phrase being safe in one context and unsafe in another—as a *judgment flip*, and treat correct handling of flips as the defining capability of a context-aware detector.

Method

Semantic context consistency model

The detector receives the structured input $\text{Input} = [\,Q\,] \,\|\, [\,C\,]$ and emits a label in $\{\text{safe}, \text{suspicious}, \text{malicious-instruction}\}$ preceded by a short reasoning trace. Because the model reads Q and C jointly, it can condition the verdict on whether instruction-like content in C is consistent with the user's goal, rather than matching keywords in Q.

Teacher→judge→student distillation

High-quality (Q, C) labels are expensive to author by hand. We instead distill judgment from larger models into the small student in three stages (Figure 1): a **teacher** generates labeled (Q, C) records; an independent, larger **judge** of a different model family re-labels each record blind to cross-check quality; and the agreed data trains the **student**.

Synthetic data generation

The teacher generates four attack families—indirect prompt injection (ipi), knowledge-base poisoning, context hijacking (role/permission redefinition), and PII exfiltration—plus benign documentation and ambiguous borderline cases (instruction-like text that is off-intent but not clearly an attack). For each adversarial record the attack payload is embedded in C (e.g., inside an HTML comment, footer, or "ignore previous instructions" clause), never in Q, forcing the model to reason about consistency rather than detect noise. Outputs use *schema-constrained decoding* (the serving engine masks any token that would violate a JSON schema), guaranteeing parseable, in-vocabulary labels.

Judgment-flip pairs. To probe context-dependent safety we additionally generate matched pairs: one imperative sentence S is placed into (a) a benign context where S is legitimate documentation (label safe) and (b) an injected context where S hijacks the assistant (label malicious-instruction). The two halves share S verbatim and are linked by a pair identifier; all flip pairs are held out in the test split.

Independent judge cross-check

A larger model from a different family re-labels every record seeing only (Q, C), blind to the teacher's label. Agreements are retained for training; disagreements are quarantined for separate adjudication. The teacher–judge agreement rate is itself a dataset-quality signal.

Reasoning-first LoRA fine-tuning

The student is fine-tuned with Low-Rank Adaptation (LoRA): the base weights are frozen and small rank-8 adapters are inserted into the attention and MLP projections. The training target is a short reasoning string followed by the label ("`Reasoning: ... Label: ...`"), with loss masked to the completion only. Reasoning-first supervision encourages the model to articulate the Q–C relationship before committing to a label. The result is a 17.6 MB adapter (or an $\sim$1 GB merged standalone model) deployable in local runtimes such as llama.cpp/Ollama.

Experimental Setup

Hardware and models

All experiments run on a single **NVIDIA DGX Spark** (GB10 Grace-Blackwell, 121 GB unified memory, aarch64, CUDA 13.0) using an NVFP4 vLLM runtime. Models run *one at a time* (the teacher and judge cannot co-reside in 121 GB):

- **Teacher:** Qwen3.6-35B-A3B (NVFP4 MoE, 3B active) — data generation.
- **Judge:** Nemotron-3-Super-120B-A12B (NVFP4 MoE, 12B active) — cross-check.
- **Student:** Qwen2.5-0.5B with LoRA (r=8, α=16, dropout 0.05; targets `q`, `k`, `v`, `o`, `gate`, `up`, `down`).

Dataset

The pipeline produced **5,233** (Q, C) records (Table 2). The independent judge agreed with the teacher on **95.0%** of records (4,973 agreed, 260 disputed, 0 without a verdict). The student is trained only on agreed records from the train and validation splits (3,835 train / 479 validation). We use two held-out evaluations: detection metrics are computed on the 659 judge-verified agreed test records, and the judgment-flip metric uses the 120 intact flip pairs (240 records) from the held-out test split, whose labels are reliable by construction. (Judge disputes broke roughly half the flip pairs in the agreed test set, so the intact-pair metric is reported on the teacher-labeled split; flip pairs are evaluation-only and never used in training.)

Table 2: Synthetic dataset composition (5,233 records).

Label	Count	Attack type	Count
safe	3,116	benign	3,115
malicious-instruction	1,917	ipi	818
suspicious	200	poisoning	500
		context-hijack	400
		pii-exfil	200
		borderline	200
Splits: train 3,994 / val 499 / test 740		Flip pairs: 120 (test only)	

Table 3: Detection on the held-out test set (659 judge-verified agreed records). ↑/↓ indicate better direction. The rightmost column is batched per-item time (throughput); single-request latency is $\sim$0.4 s (Sec.).

System	Recall ↑	ASR-proxy ↓	FPR ↓	3-class acc ↑	batch p50
B1 regex (context-blind)	0.688	0.312	0.129	0.777	$<$1 ms
B2 Qwen2.5-0.5B (zero-shot)	0.116	0.884	0.059	0.533	55 ms
B3 tuned SLM (ours)	**0.994**	**0.006**	**0.026**	**0.974**	**37 ms**

Baselines

We compare three systems: **B1** a regex/heuristic filter scanning C for injection patterns (context-blind); **B2** the base Qwen2.5-0.5B used *zero-shot* with the same prompt (no fine-tuning); and **B3** our LoRA-tuned SLM. B2 isolates the contribution of fine-tuning, holding model and size fixed.

Metrics

Detection Recall (fraction of malicious caught), **Attack Success Rate (ASR)** as a detection-side proxy (fraction of malicious not flagged), **False Positive Rate (FPR)** (benign incorrectly flagged), three-class accuracy, and end-to-end **latency**. For the flip set we report per-record accuracy and the stricter *both-correct* rate (both halves of a pair labeled correctly).

Reproducibility

All stages use a fixed random seed (20,260,624) with frozen train/validation/test splits; the teacher, judge, and base student use pinned NVFP4 / base-model revisions served via vLLM. The model, synthetic-data generation pipeline, and evaluation harness are released to support replication.

Results

Detection performance

Table 3 reports the 659 judge-verified agreed test records. The tuned SLM (B3) dominates every security metric: 0.994 recall, 0.006 ASR-proxy, 0.026 FPR, 0.974 three-class accuracy. The *same* model used zero-shot (B2) reaches only 0.116 recall—it collapses to predicting safe for the large majority of inputs—so fine-tuning raises recall from 0.116 to 0.994. The regex baseline (B1) catches 68.8% of attacks but over-flags benign text (0.129 FPR).

Judgment-flip evaluation

Table 4 reports the 120 intact flip pairs. The tuned SLM labels both halves of a pair correctly 75.0% of the time, versus

Table 4: Judgment-flip subset (120 intact pairs / 240 records).

System	flip-record acc ↑	flip-pair both-correct ↑
B1 regex	0.588	0.175
B2 Qwen2.5-0.5B (zero-shot)	0.529	0.117
B3 tuned SLM (ours)	**0.875**	**0.750**

17.5% for the regex baseline and 11.7% for the untuned model. Because a keyword filter assigns the same label to the shared sentence S regardless of context, it is structurally incapable of solving flips; the context-aware model is not. This result is the most direct evidence for the paper's central claim.

Latency and efficiency

We distinguish throughput from latency. Under batched evaluation (batch 32) the tuned SLM sustains 37 ms per (Q, C) amortized—a *throughput* figure. The deployment-relevant *single-request* latency is 404 ms median (513 ms p95) in eager transformers, dominated by generating the $\sim$36-word reasoning trace before the label; this exceeds the interactive sub-100 ms target. A label-only variant (dropping the reasoning trace) or an optimized serving runtime (vLLM/llama.cpp) is the path to sub-100 ms single-request latency, which we leave to future work. End-to-end the pipeline is nonetheless cheap: data generation drew $\sim$35 Wh, fine-tuning $\sim$9 Wh (14.6 minutes, 94% GPU utilization), and the final adapter is 17.6 MB. The entire teacher→judge→student pipeline runs on a single consumer-class device.

Data quality

The 95.0% teacher–judge agreement indicates the labels are largely trustworthy while the 5% disputed set (quarantined, not silently trusted) concentrates on the genuinely ambiguous suspicious boundary. Critically, the regex baseline catches only 68.8% of attacks and the untuned model only 11.6%, confirming the data is not trivially separable—so the tuned model's performance reflects learned capability rather than an easy benchmark.

Discussion

The experiments support three claims. First, **context matters**: reframing the task over (Q, C) rather than Q enables a small model to resolve judgment flips that are by construction unsolvable for keyword filters. Second, **distillation works at small scale**: a 0.5B student trained on teacher/judge-labeled synthetic data inherits judgment far beyond its zero-shot ability ($0.116 \rightarrow 0.994$ recall). Third, **the defense is local and cheap**: it meets interactive latency and tiny-footprint targets without any cloud guard, satisfying the privacy motivation of local RAG. The independent cross-model judge provides a quantitative quality gate (95% agreement) that strengthens confidence in synthetic-data labels.

Limitations

We state limitations explicitly. (1) **In-distribution.** Training and test data share the same synthetic generator; the strong numbers are in-distribution and do not establish robustness on external benchmarks (e.g., OpenRAG-Soc (Guo and Wei 2026)) or in the wild. (2) **ASR is a detection-side proxy**—whether the guard flags the attack—not the downstream generator's behavior. (3) **Adaptive and optimization-based adversaries are untested.** We have not evaluated gradient-based prompt injection, adaptive multi-chunk or fragmented injections, retrieval-optimized attacks that craft passages to guarantee retrieval (Chang et al. 2026; Chan et al. 2024), or human red-team attacks; the in-distribution numbers do not speak to these. (4) Latency is measured under batched eager inference on one device and will vary with runtime and hardware. (5) Data is English-only. (6) **Single-domain validation.** Our synthetic corpus is general-purpose; we have not yet validated on financial, medical, or legal documents, where dense instruction-like text and domain jargon may shift the safe/malicious boundary. Cross-domain and external-benchmark (e.g., SafeRAG (Liang et al. 2025)) validation is a priority for the next iteration. We therefore present this as evidence that the approach is viable and promising, not as a claim of deployed-grade robustness.

Conclusion

We implemented and evaluated nokast-secureRAG, a local $\sim$0.5B semantic firewall for indirect prompt injection in RAG, produced by a teacher→judge→student distillation pipeline that runs entirely on a single consumer-class device. The tuned SLM reaches 0.994 recall at 0.026 FPR, and—most importantly—solves context-dependent "judgment flip" cases that keyword baselines cannot. External-benchmark generalization, true downstream-ASR measurement, adaptive-adversary stress testing, and sub-100 ms single-request latency are the focus of the next iteration of this work. The model, synthetic-data pipeline, and evaluation harness are released open source.

Ethics Statement

This work addresses security vulnerabilities in AI systems and aims to improve the safety of local RAG deployments. The synthetic data generation pipeline and evaluation harness are released open source to support reproducibility and further research. We acknowledge that our defense is in-distribution validated and does not claim robustness against adaptive adversaries, which we discuss explicitly in the Limitations section.

References

Bassani, E.; and Sanchez, I. 2024. GuardBench: A large-scale benchmark for guardrail models. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

Chan, W.; Li, Z.; Sun, Y.; Zhang, Y.; and Chen, Y. 2024. Red-teaming LLM agents via poisoning memory or knowledge bases. *arXiv preprint arXiv:2407.12784*.

Chang, H.; Bao, E.; Luo, X.; and Yu, T. 2026. Overcoming the retrieval barrier: Indirect prompt injection in the wild for LLM systems. *arXiv preprint arXiv:2601.07072*.

Greshake, K.; Abdelnabi, S.; Endres, C.; Holz, T.; Sunyaev, A.; and Fritz, M. 2023. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*.

Guardrails AI. 2024. Guardrails AI: An open-source framework for building reliable AI applications.

Guo, H.; and Wei, Z. 2026. Hidden-in-plain-text: A benchmark for social-web indirect prompt injection in RAG. *arXiv preprint arXiv:2601.10923*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning Workshop*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-rank adaptation of large language models. In *Proc. Int. Conf. Learning Representations (ICLR)*.

IBM Security. 2024. Cost of a data breach report 2024.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; and Khabsa, M. 2023. Llama Guard: LLM-based input–output safeguard for human–AI conversations. *arXiv preprint arXiv:2312.06674*.

Khodayari, S.; Zhang, X.; Acharya, B.; and Pellegrino, G. 2026. Indirect prompt injection in the wild: An empirical study of prevalence, techniques, and objectives. *arXiv preprint arXiv:2604.27202*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Liang, X.; Niu, S.; Li, Z.; Zhang, S.; Wang, H.; Xiong, F.; Fan, J.; Tang, B.; Song, S.; Wang, M.; and Yang, J. 2025. SafeRAG: Benchmarking security in retrieval-augmented generation of large language model. *arXiv preprint arXiv:2501.18636*.

Liu, Y.; Deng, G.; Li, Y.; Wang, K.; Wang, Z.; Wang, X.; Zhang, T.; Liu, Y.; Wang, H.; Zheng, Y.; and Liu, Y. 2023a. Prompt injection attack against LLM-integrated applications. *arXiv preprint arXiv:2306.05499*.

Liu, Y.; Jia, Y.; Geng, R.; Jia, J.; and Gong, N. Z. 2023b. Formalizing and benchmarking prompt injection attacks and defenses. *arXiv preprint arXiv:2310.12815*.

Perez, F.; and Ribeiro, I. 2022. Ignore previous prompt: Attack techniques for language models. In *NeurIPS ML Safety Workshop*.

Rebedea, T.; Dinu, R.; Sreedhar, M.; Parisien, C.; and Cohen, J. 2023. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*.

She, Y.; Peterson, D. W.; Liu, M.; Upadhyay, V.; Chaghazardi, M. H.; Kang, E.; and Roth, D. 2025. RAG makes guardrails unsafe? Investigating robustness of guardrails under RAG-style contexts. *arXiv preprint arXiv:2510.05310*.

Unit 42, Palo Alto Networks. 2026. Fooling AI agents: Web-based indirect prompt injection observed in the wild.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yi, J.; Xie, Y.; Zhu, B.; Kiciman, E.; Sun, G.; Xie, X.; and Wu, F. 2023. Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv preprint arXiv:2312.14197*.

Zhang, B.; Xin, H.; Li, J.; Zhang, D.; Fang, M.; Liu, Z.; Nie, L.; and Liu, Z. 2025. Benchmarking poisoning attacks against retrieval-augmented generation. *arXiv preprint arXiv:2505.18543*.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Zou, W.; Geng, R.; Wang, B.; and Jia, J. 2024. PoisonedRAG: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.