Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.38107

Indirect Prompt Injection Attacks on LLM Agents and Defense Mechanisms

Zijia Guo

School of Computer Science, Shandong Xiehe University, Jinan, China
G139534077955@outlook.com

Abstract. LLM agents are becoming deeply involved in real-world tasks through tool calling and external retrieval; however, by merely contaminating the external content read by these agents, such as web pages and emails, attackers can remotely hijack their execution logic. This threat is known as indirect prompt injection (IPI). This paper presents a systematic survey of IPI research on LLM agents from three perspectives: attacks, defenses, and evaluation. On the attack side, a three-layer taxonomy of attack entry, attacker goals, and attack outcomes is established, covering seven categories of entry paths, five categories of hijacking mechanisms, and five categories of harms. On the defense side, existing defenses are organized by their point of intervention into a five-layer system spanning the input, processing, behavior, training, and output layers, and their effectiveness and deployment costs are compared horizontally. In addition, representative evaluation benchmarks such as BIPIA and InjecAgent are reviewed, and a three-dimensional evaluation metric system covering the attack side, the defense side, and the system side is proposed. The survey shows that existing defenses significantly improve robustness against direct prompt injection, whereas the improvement against indirect prompt injection is at most 7.5 percentage points, and every single-layer defense has blind spots. Multi-layer coordinated defense-in-depth and a unified evaluation framework are therefore key directions for future research.

Keywords: LLM agents, indirect prompt injection (IPI), prompt injection defense, retrieval-augmented generation (RAG), tool calling

1. Introduction

As large language models (LLMs) evolve from conversational tools into autonomous actors, LLM agents have been granted operational privileges that affect the real world, such as retrieving web pages, reading emails, invoking APIs, and generating code [1]. However, to accomplish their tasks, these agents must read external content, which can be contaminated by attackers. Without interacting directly with the target system, an attacker only needs to implant malicious instructions in advance into web pages, emails, documents, and other carriers; once read by the agent, these instructions are ingested along with the context and executed. This threat is referred to as indirect prompt injection (IPI).

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38107

Unlike direct prompt injection, no direct communication channel exists between the IPI attacker and the target system, rendering this attack pattern highly stealthy. Greshake et al. first demonstrated the feasibility of such attacks through systematic experiments on real-world LLM-integrated applications, revealing that these attacks are low-cost to launch and difficult to detect [2].

Existing research on IPI remains fragmented. Attack taxonomies involve overlapping concepts and lack a unified classification logic, while defense research lags behind attack research, with little horizontal comparison across methods. The problems in evaluation are even more pronounced: studies vary widely in their datasets, experimental setups, and metrics, rendering their results difficult to compare fairly. Overall, the research community still lacks a comprehensive technical landscape covering attacks, defenses, and evaluation.

To address these issues, this paper presents a systematic survey of IPI attacks and defenses for LLM agents. Specifically, it establish a three-layer attack taxonomy of attack entry, attacker goals, and attack outcomes; organize existing defenses into a five-layer system spanning the input, processing, behavior, training, and output layers; propose a three-dimensional evaluation metric system covering the attack side, the defense side, and the system side; and discuss the core challenges and future directions of this field.

2. Indirect prompt injection attack methods

This chapter organizes existing attack methods into a three-layer framework of attack entry, attacker goals, and attack outcomes, along three dimensions: how malicious instructions penetrate, which component they hijack, and what harm they cause.

From the perspective of attack entry, the penetration channels for malicious instructions are highly diverse. Conventional entry paths include passive embedding in public web pages and technical documents, active delivery via emails and instant messages, and social engineering that induces users to execute the instructions [2, 3]. As system architectures grow more complex, malicious content can further hide in code repositories, retrieval-augmented generation (RAG) knowledge bases, multi-modal media files, and even the user's long-term conversational memory, and is ultimately ingested by the agent together with external inputs [2, 4].

From the perspective of attacker goals, hijacking targets multiple components inside the agent, covering the instruction parsing process, the tool calling chain, the code synthesis chain, the knowledge fusion process, and the browser interaction chain. Attackers exploit the model's default trust in external content and tool-returned data to perform unauthorized operations, with an empirically measured attack success rate (ASR) of up to 62.5% [2-5].

From the perspective of attack outcomes, the harms span a complete spectrum from immediate damage to persistent control, including direct attacks, private data exfiltration, remote control, chained propagation, and privilege escalation [2-4].

The three layers are not isolated from one another but are connected sequentially along the same attack chain. The entry paths exploit the agent's passive ingestion of external content, allowing attackers to embed payloads in advance without any contact with the target system. The hijacking mechanisms, in turn, are enabled by the model's inherent trust in retrieved content and tool-returned data. As for attack outcomes, their severity scales with the agent's privileges, ranging from immediate damage to persistence, and even to cascading spread across sessions and users. Entry, goals, and outcomes can be freely combined: any entry path may propagate through different hijacking mechanisms to produce multiple harms, which is precisely why IPI defenses must cover the entire attack chain.

DOI: 10.54254/2755-2721/2026.38107

3. Existing defense mechanisms

This chapter reviews existing defense mechanisms against IPI for LLM agents. The analysis in Section 2 shows that IPI attacks unfold along the entry–goal–outcome chain; accordingly, defense research has formed multiple lines of defense at the different stages where agents process external content. As shown in Figure 1, this chapter organizes existing methods into five layers—the input, processing, behavior, training, and output layers—according to their point of intervention. The first four layers have all been experimentally validated; however, the training layer intervenes at the model training stage and thus applies only to deployment scenarios with fine-tuning access. The output layer is a different case: no specifically validated IPI defense method exists at this layer to date, so this paper positions it as a supplementary verification stage that takes effect after the preceding layers fail, and discusses it only as a research concept.

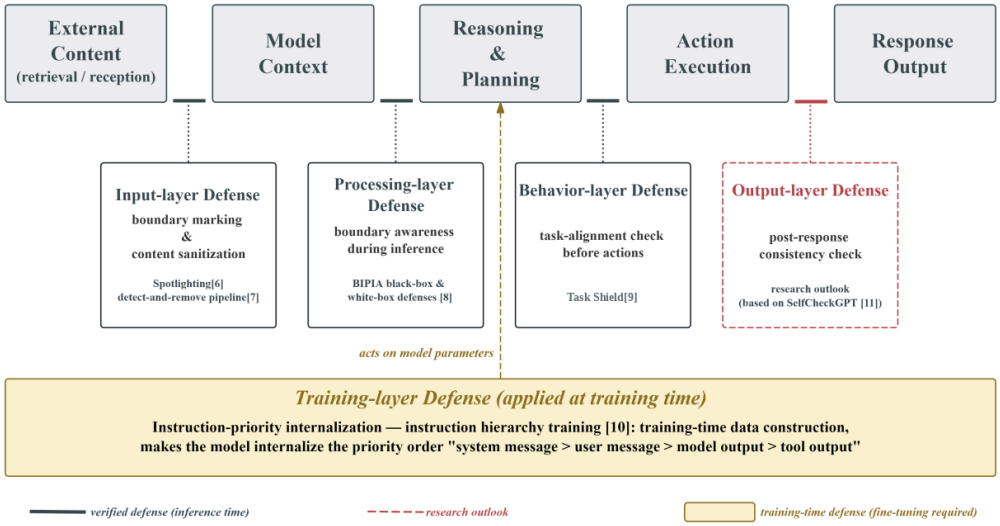


Figure 1. Five-layer defense framework against indirect prompt injection (photo/picture credit: original)

3.1. Input-layer defense: boundary explicitness and sanitization of external content

Input-layer defenses intervene before external content enters the model and unfold along two paths. Spotlighting makes the boundaries of external content explicit within the prompt through delimiting, datamarking, or encoding, enabling the model to distinguish such content from system instructions [6]. Chen et al. examined the sanitization path of detection and removal, systematically comparing four categories of post-hoc defenses, among which segment-based removal—splitting the input into sentences, judging each segment, and removing injected fragments—is the most representative [7].

3.2. Processing-layer defense: boundary awareness during inference

Processing-layer defenses act at the model inference stage; their core idea is to keep the model aware of the boundary between instructions and data while it processes external content. The attribution analysis by Yi et al. attributes indirect injection to the model's lack of provenance awareness and its tendency to follow instructions in the concatenated input verbatim [8]. Building on this analysis, the black-box defenses distinguish instructions from data through multi-turn dialogue and in-context learning, while the white-box defense strengthens boundary awareness by

DOI: 10.54254/2755-2721/2026.38107

adding special tokens (e. g., <data></data>) to mark external content during fine-tuning; both significantly reduce the ASR on the BIPIA benchmark [8]. Notably, the study observed that more capable models are instead more susceptible to injection due to their stronger instruction-following tendency, which imposes additional requirements on the protection of high-capability models [8]. Ablation experiments further show that explicitly recognizing the boundary tokens is critical to the effectiveness of the defense [8].

3.3. Behavior-layer defense: task alignment verification of action decisions

Behavior-layer defenses focus on whether the agent's action decisions are aligned with the user's initial task. Since malicious instructions can be disguised as semantically plausible and thus evade boundary awareness, Task Shield, proposed by Jia et al., shifts the defense criterion: instead of enumerating the features of harmful content, it verifies before each tool call whether the call serves the user's instruction [9]. Its limitation is that the verification relies on the model's own judgment; when the model has been misled by injected content, the reliability of the verification degrades accordingly [9].

3.4. Training-layer defense: the internalization of instruction priorities

The aforementioned three layers are all inference-time defenses, which cannot address the root cause: the model treats system instructions, user inputs, and external content as equally trustworthy. The instruction hierarchy training proposed by Wallace et al. moves the defense forward to the training stage; by constructing training data that distinguishes instruction priorities and applying fine-tuning, the model is made to internalize the hierarchy of instruction priorities [10]. As shown in Figure 2, this method significantly improves robustness against direct injection attacks, whereas the improvement against indirect injection via browsing is limited, indicating that data-driven priority learning cannot yet fully cover the complex injection patterns found in external content [10].

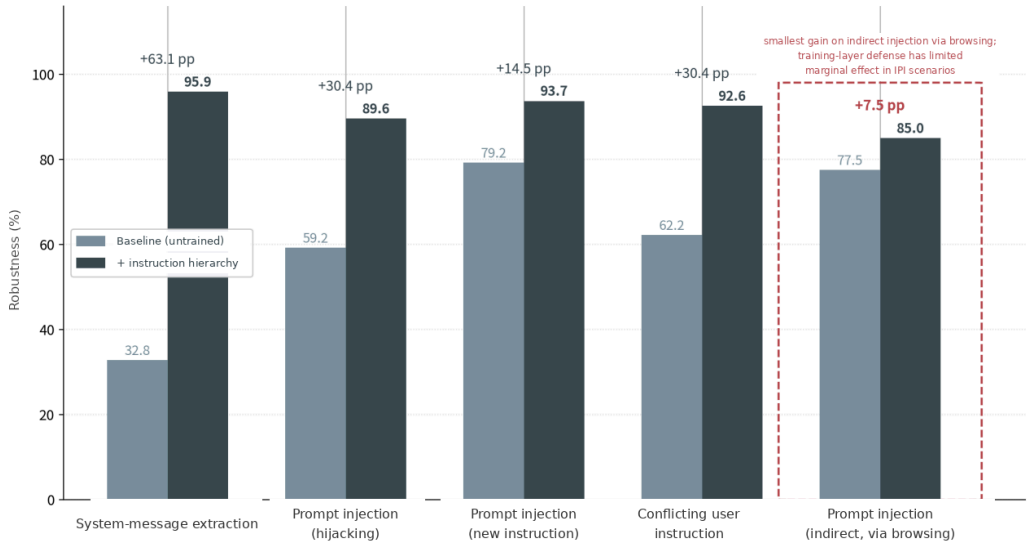


Figure 2. Robustness improvement of instruction hierarchy training across attack types [10]

DOI: 10.54254/2755-2721/2026.38107

3.5. Output-layer defense: consistency checking as an exploratory direction

Within the scope of this survey, no specifically validated IPI defense method exists at the output layer, and this paper positions it as a research direction worth exploring. In a neighboring field, SelfCheckGPT, proposed by Manakul et al., offers a transferable idea for hallucination detection: sampling multiple outputs for the same input and checking the consistency among them [11]. If transferred to output-side defense, an inconsistency between the agent's final output and its historical behavior trajectory or task goal may indicate that external content has interfered with the decision-making process.

3.6. Horizontal comparison of defense mechanisms and analysis of their advantages and disadvantages

Table 1. Horizontal comparison of six categories of defense mechanisms

Defense method	Empirical effectiveness and utility	Deployment requirements	Main limitations
Spotlighting [6] (input layer)	ASR reduced from >50% to <2%, with negligible utility impact	Deployable by only adjusting the system prompt	Delimiters are easy to forge; encoding depends on model capability
Detection-and-removal pipeline [7] (input layer)	Depends on detector accuracy; may break document coherence	Requires training a dedicated detection model	Cross-domain overkill; poor generalization to injection positions
BIPIA black-box [8] (processing layer)	Significantly mitigates attacks with almost no impact on task utility	Usable in black-box settings; no model modification needed	Relies on prompt design; does not address the root of trust
BIPIA white-box [8] (processing layer)	Effectively blocks attacks without degrading output quality	Requires fine-tuning access to an open-source model	Limited to white-box scenarios
Task Shield [9] (behavior layer)	ASR reduced to 2.07% with 69.79% utility retained	An extra LLM call per round; high cost	Depends on the underlying LLM's capability; fails in open-ended dialogue
Instruction hierarchy [10] (training layer)	+63.1 pp on direct injection but only +7.5 pp on indirect injection; over-refusals observed	Requires a full training pipeline; open-source models only	Marginal gain on indirect injection; fixed hierarchy assumption

The five layers above cover the complete processing pipeline from input to output; a horizontal comparison of six representative methods and the advantages and disadvantages of each layer are summarized in Table 1 and Table 2, respectively. Taken together, no single-layer defense covers the entire attack surface, and multi-layer coordinated defense-in-depth is the realistic choice. There is also a clear trade-off between defense depth and deployment cost: methods that are easy to deploy offer limited protection depth, while deeper protection comes at the cost of model access privileges or training costs. Moreover, the evaluation settings of existing defenses are highly heterogeneous:

DOI: 10.54254/2755-2721/2026.38107

the ASR values reported by different methods come from different benchmarks and different attack implementations, so the numbers should not be directly compared horizontally. This issue is discussed further in Section 4.

Table 2. Comparison of the advantages and disadvantages of the five defense layers

Layer	Advantages	Disadvantages
Input layer	Usable in black-box settings; easy to deploy	Limited detection rate against stealthy attacks
Processing layer	Defense proceeds in sync with inference; significant effectiveness	The white-box scheme requires internal access to the model
Behavior layer	Independent of model internals; broadly applicable	Verification reliability is affected by the model's own state
Training layer	Durable defense; resists multiple attack variants	Relies on model providers; hard for ordinary developers to adopt
Output layer	Post-hoc checking; in theory can cover attacks missed by all previous layers	Only a research concept so far; no mature method

4. Evaluation benchmarks and metric system

4.1. Evaluation benchmarks and injection construction methods

Existing IPI evaluation benchmarks fall into three categories by application scenario: QA benchmarks for general text tasks, interactive benchmarks for tool-integrated agents, and specialized benchmarks for specific architectures; their task compositions, attack implementations, and measurement criteria are compared in Table 3. BIPIA is the first cross-model quantitative benchmark in this field, covering five categories of QA tasks—email, web page, table, summarization, and code [8]. InjecAgent provides single-turn static test cases, whereas AgentDojo advances evaluation to multi-turn dynamic interaction, formally determining whether an attack achieves its goal based on changes in environment state [3, 12]. In addition, DeceptPrompt targets code generation models [4], and PANDORA targets RAG-enhanced GPT applications, representing benchmark designs for specific architectures [5].

Table 3. A comprehensive comparison of IPI evaluation benchmarks

Benchmark (scenario)	Task composition and attack implementation	Measurement criteria
BIPIA [8] (general text tasks)	5 task categories; 250 attacker goals; 30 text attacks + 20 code attacks; adjustable injection positions; evaluated on 25 LLMs	ASR + benign task performance
InjecAgent [3] (tool-integrated agents, single-turn static)	17 user tools × 62 attacker tools; 62 attack cases; two settings (basic injection and hacking-prompt enhancement); 30 agents and 1,054 cases in total	ASR (judged by whether the attacker tool is invoked and causes harm)

DOI: 10.54254/2755-2721/2026.38107

Table 3. (continued)

AgentDojo [12] (tool-integrated agents, multi-turn dynamic)	4 simulated environments; 97 user tasks; an extensible library of attack paradigms; injection tasks crossed with user tasks; 629 security test cases; covering models such as GPT-4o and the Claude family	utility + targeted ASR (formally judged by environment state)
DeceptPrompt [4] (code generation)	Code generation tasks + adversarial prefixes; semantically benign prefixes generated by an evolution-based algorithm; evaluated on code models such as CodeLlama-7B	ASR + vulnerable-function occurrence rate + perplexity
PANDORA [5] (RAG/GPTs)	QA in four forbidden-content scenarios; knowledge-base-poisoned documents; evaluated for GPT-3.5 and GPT-4	Jailbreak success rate

Beyond unified benchmarks, the way attack implementations are constructed also needs standardization. By construction method, Chen et al. divide injection payloads into five categories—Naive, Ignore, Escape, Fake Completion, and Combined, i. e., naive injection, overwrite injection, escape injection, fake-completion injection, and combined injection—and point out that detectors trained only on naive injection generalize poorly to attacks with more complex constructions [7]. Therefore, evaluation reports must specify the injection construction method used; otherwise, conclusions about ASR reduction lack a comparable basis.

4.2. A three-dimensional evaluation metric system

Building on the above benchmark inventory, this paper organizes the measurement practices scattered across the literature into a three-dimensional evaluation metric system covering the attack side, the defense side, and the system side, as shown in Table 4. It should be noted that core metrics such as ASR are systematically affected by experimental conditions, including the task context, the injection field type, and how the agent is constructed; the success rate of the same attack can differ severalfold under different conditions. This system therefore requires that ASR reports specify these variables as well; otherwise, the values are not horizontally comparable. The attack side answers how strong an attack is, with ASR as the core metric; it recommends adopting AgentDojo's criterion of goal achievement rate, i. e., judging success by whether the attacker's predefined goal is fully realized in the environment state, to avoid inflated values caused by cases where the injection takes effect but the goal is not achieved [12]. Payload stealthiness serves as a supplementary metric: DeceptPrompt's practice shows that a highly stealthy payload can achieve an ASR of 62.5% while its perplexity is almost indistinguishable from that of normal text [4]. The defense side answers how effective a defense is, centered on the ASR reduction under the same benchmark and the same attack implementation, while also reporting the task utility retention rate and, for detection-based defenses, the detection rate and false positive rate; Spotlighting and Task Shield both achieve order-of-magnitude ASR reductions on their respective benchmarks [6, 9]. The system side answers how costly a defense is, covering additional invocation costs, latency overhead, and deployment privilege requirements; deployment privileges fall into three levels—black-box usability, white-box fine-tuning privileges, and full training pipeline access—corresponding to Spotlighting, the BIPIA white-box defense, and instruction hierarchy training, respectively [6, 8, 10].

DOI: 10.54254/2755-2721/2026.38107

Table 4. The three-dimensional evaluation metric system

Dimension	Metric	Definition and reporting requirements
Attack side	Attack success rate (ASR)	Number of successful attack cases ÷ total test cases; state the success criterion (lenient/strict)
	Goal achievement rate (targeted ASR)	Proportion of cases in which the attacker's goal is fully realized; formally judged by environment state or behavior logs
	Payload stealthiness	Perplexity (PPL) and semantic similarity to normal text; state the scoring model and threshold used
Defense side	ASR reduction	ASR before defense − ASR after defense (in percentage points); measured under the same benchmark and the same attack implementation
	Task utility retention rate	Benign task completion rate after defense ÷ original completion rate; reported jointly with ASR reduction
	Detection rate and false positive rate	TPR / FPR or F1 (applicable to detection-based defenses); state whether cross-domain and cross-attack-type
	Additional invocation cost	Additional LLM calls or token overhead per task; state the number of samples and the model size
System side	Latency overhead	End-to-end latency increment introduced by the defense; state the test environment
	Deployment privilege requirements	Three levels: black-box / white-box fine-tuning / full training access

This metric system is not an empty framework but a yardstick already applied in the comparative analysis of this paper. The column structure of Table 1 in Section 3, which horizontally compares the six defense mechanisms, is an instantiation of this system: the column of empirical effectiveness and utility corresponds to the defense-side metrics and the utility retention rate, and the column of deployment requirements corresponds to the privilege dimension of the system side.

At this point, this paper has completed its review of attack methods, defense mechanisms, and the evaluation system. The three-dimensional metrics together reveal the asymmetry of the current attack–defense landscape. On the attack side, the barrier continues to lower: payloads can be embedded without any direct interaction with the target system [2], and their construction can combine high stealthiness with cross-model transferability [4, 7]. On the defense side, methods are generally constrained by the trade-off between security and utility, and no single method yet achieves high ASR reduction, low overhead, and zero utility loss simultaneously [6, 9]. The underlying causes of this landscape and possible breakthrough directions are discussed in Section 5.

DOI: 10.54254/2755-2721/2026.38107

5. Challenges and future directions

5.1. Core challenges facing current research

The most fundamental challenge remains at the architecture level. LLMs process all inputs within a single sequence modeling framework; unlike traditional computing systems, there is no hard boundary of separated storage between instructions and data, so the model cannot structurally refuse to execute commands arriving through the data channel. The gain of only 7.5 percentage points from training-layer defense against indirect injection likewise shows that existing alignment techniques have not yet established a reliable trust hierarchy inside the model [10]. Agent pipelines amplify this flaw: retrieved results are regarded as trustworthy knowledge by default and passed along the retriever–model–tool chain, with no stage reviewing the upstream content [5]. As long as this architectural premise remains unchanged, defenses can only mitigate the threat rather than eradicate it.

The second challenge lies at the measurement level. Existing benchmarks are essentially static snapshots of attack methods, whereas payload construction keeps evolving. Even AgentDojo, the most recent benchmark in interaction form, has its task set and attack implementations fixed on the day of release, and evaluation conclusions drawn from such benchmarks quickly become outdated as attacks evolve [12]. Meanwhile, benchmarks differ in their criteria for judging defense success, and utility retention and invocation costs lack unified measurement, making attack–defense progress difficult to compare horizontally.

The third challenge comes from the expansion of the attack surface. As agents connect to multi-modal tools, embodied environments, and even other agents, injection carriers are expanding from plain text to images, audio, and sensor data, and the trust chain extends across agents: one agent's output may become another agent's input, giving injections a path to propagate across systems. However, existing attack and defense research focuses almost entirely on plain-text scenarios, and there is no systematic understanding of the mechanisms of, or protection against, these new carriers.

5.2. Prospects for future research directions

Viewed by implementation actor and maturity, the foreseeable paths are roughly as follows. What application developers can work on in the near term is output-layer defense. Section 3.5 has discussed the feasible path of transferring consistency checking to IPI defense [11]. It should be added that this checking acts on the final output rather than on intermediate decisions, and therefore forms defense-in-depth together with the behavior-layer defense that intercepts before decisions are made [9]. Its practical deployment must also face the false-positive cost specific to agent scenarios: a single false positive may interrupt an entire legitimate task chain.

The community's medium-term task is to build dynamically evolving evaluation benchmarks. Attack implementations and task sets need to be continuously updated by red teams, so that evaluation conclusions can keep pace with attack evolution. The three-dimensional metric system proposed in Section 4.2 can serve as a starting point for a unified reporting standard, making the ASR, utility, and cost of different defenses comparable.

Longer-term work depends on model providers. The first direction is to explore new architectures that separate instructions from data—for example, introducing privilege tags or channel isolation at the input side—so that the model distinguishes commands from content at the source. The second is to lower the implementation barrier of instruction hierarchy training, so that application developers

DOI: 10.54254/2755-2721/2026.38107

without full training resources can also inherit training-layer protection [10]. Injection defenses for multi-modal and multi-agent scenarios await further development once the above directions mature.

6. Conclusion

Indirect prompt injection targeting LLM agents has evolved from proof of concept into a real-world threat, yet related research has long remained fragmented: attack taxonomies lack a unified logic, defense methods lack horizontal comparison, and evaluation criteria vary. To address this situation, this paper presents a systematic survey of IPI attack and defense research for LLM agents, establishes a three-layer taxonomy of attack entry, attacker goals, and attack outcomes, organized existing defenses by their point of intervention into a five-layer system spanning the input, processing, behavior, training, and output layers, and proposes a three-dimensional evaluation metric system covering the attack side, the defense side, and the system side.

The survey yields two core findings. In terms of attack and defense, IPI attacks have a low barrier yet a complete spectrum of harms: mixing only about 1.8% of malicious tokens into retrieved content suffices to carry out an injection, and the attack success rate against prompted agents reaches up to 47%. By contrast, existing defenses improve robustness against indirect injection by at most 7.5 percentage points; every single-layer defense has blind spots, and the attack–defense landscape is markedly asymmetric. In terms of evaluation, differences in task settings and judgment criteria across benchmarks make most reported values difficult to compare horizontally; measured by the unified three-dimensional metrics, no single defense yet achieves high ASR reduction, low overhead, and zero utility loss simultaneously.

These findings have dual value. Theoretically, the three-layer framework and the three-dimensional metrics provide a unified technical landscape and a comparative yardstick for fragmented IPI research. Practically, the horizontal comparison of the five defense layers offers direct guidance for agent developers in selecting and deploying multi-layer coordinated defenses. It can be expected that architectural innovations separating instructions from data, together with dynamically evolving evaluation benchmarks, will be the key directions for breaking through the current limitations of defenses.

References

[1] Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., & Scialom, T. (2023). Augmented language models: A survey. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=jh7wH2AzKK

[2] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security* (pp. 79–90). Association for Computing Machinery. https://doi.org/10.1145/3605764.3623985

[3] Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024* (pp. 10471–10506). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.624

[4] Deng, G., Liu, Y., Wang, K., Li, Y., Zhang, T., & Liu, Y. (2024). PANDORA: Jailbreak GPTs by retrieval augmented generation poisoning. In *Proceedings of the Workshop on AI Systems with Confidential Computing (AISCC 2024)*. Internet Society. https://doi.org/10.14722/aiscc.2024.23018

[5] Wu, F., Liu, X., & Xiao, C. (2023). DeceptPrompt: Exploiting LLM-driven code generation via adversarial natural language instructions. arXiv. https://doi.org/10.48550/arXiv.2312.04730

[6] Hines, K., Lopez, G., Hall, M., Zarfati, F., Zunger, Y., & Kıcıman, E. (2024). Defending against indirect prompt injection attacks with spotlighting. arXiv. https://doi.org/10.48550/arXiv.2403.14720

DOI: 10.54254/2755-2721/2026.38107

[7] Chen, Y., Li, H., Sui, Y., He, Y., Liu, Y., Song, Y., & Hooi, B. (2025). Can indirect prompt injection attacks be detected and removed? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 18189–18206). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.acl-long.890

[8] Yi, J., Xie, Y., Zhu, B., Kiciman, E., Sun, G., Xie, X., & Wu, F. (2025). Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1* (pp. 1809–1820). Association for Computing Machinery. https://doi.org/10.1145/3690624.3709179

[9] Jia, F., Wu, T., Qin, X., & Squicciarini, A. (2025). The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 29680–29697). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.acl-long.1435

[10] Wallace, E., Xiao, K., Leike, R., Weng, L., Heidecke, J., & Beutel, A. (2024). The instruction hierarchy: Training LLMs to prioritize privileged instructions. arXiv. https://doi.org/10.48550/arXiv.2404.13208

[11] Manakul, P., Liusie, A., & Gales, M. (2023). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (pp. 9004–9017). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.557

[12] Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., & Tramèr, F. (2024). AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems* (Vol. 37, pp. 82895–82920). Curran Associates, Inc.