NOT JUST DETECTION: ALIGNED-DRIVEN PURIFICATION OF INDIRECT PROMPT INJECTION FOR RELIABLE AGENT INTERACTION

Qingxin Xiao[⋆†] *Boyang Zhang*[⋆†] *Ziqian Zeng*[‡] *Zhengle Ding*[⋆†] *Xin Cheng*[⋆]
Lingwei Dang[⋆] *Jinhui Wu*[⋆†] *Yangyang Zhao*[§] *Qingyao Wu*[⋆‖*]

[⋆] South China University of Technology, School of Software Engineering, China
[†]Institute for Super Robotics (Huangpu), China [‖]Pengcheng Laboratory, Shenzhen, China
[‡]South China University of Technology, Shien-Ming Wu School of Intelligent Engineering, China
[§]Changsha University of Science and Technology,
School of Computer and Communication Engineering, China

ABSTRACT

This paper proposes an innovative defense framework against indirect prompt injection attacks in agent systems, integrating structure-aware attention-based detection with preference-aligned purification. Our method effectively identifies and removes malicious instructions embedded within structured interaction data, while preserving task utility and ensuring model security. Specifically, we design an end-to-end structured defense pipeline that combines supervised fine-tuning and reinforcement-based policy optimization to accurately filter adversarial content without compromising structural integrity. To support the training of the purification module, we construct the first adversarial dataset tailored to structured indirect injection scenarios. Furthermore, we introduce a novel attack variant that manipulates response data fields to simulate more deceptive and realistic threats targeting agent behavior. Experimental results on the AgentDojo benchmark demonstrate that, compared to existing detection-based defenses, our method not only significantly reduces attack success rates but also substantially improves the agent's task completion performance in interactive settings. Project page: `https://ilianzby.github.io/Agent-IPI-SID-Defense/`.

***Index Terms*—** Indirect Prompt Injection, Agent Safety, Group Relative Policy Optimization, Supervised Fine-Tuning

1. INTRODUCTION

The widespread deployment of Large Language Models (LLMs) in agent systems has significantly enhanced capabilities in question answering, task planning, and complex instruction execution. However, their high sensitivity to prompts also introduces critical security risks. In particular, **Prompt Injection (PI)** [1] attacks have emerged as a key threat to the stability and success rate of agent-based interactions in open environments.

PI attacks fall into two categories: direct and indirect. **Direct PI (DPI)** [2] occurs when an attacker inserts malicious instructions directly into user inputs. In contrast, **Indirect PI (IPI)** [3] is more insidious, as attackers embed harmful instructions into environmental content such as web pages, documents, logs, or API responses. These malicious inputs are then indirectly introduced into the model's context via toolchain calls, leading the LLM to execute unauthorized commands. In real-world deployments, where agents frequently interact with open and dynamic environments, indirect PI attacks are significantly more common. Nevertheless, most existing research has focused on the direct PI setting [4], leaving the more universal and stealthy indirect scenario largely underexplored [5].

Ignore Attack

{"meeting_id": "2023-Q4-OKR",
"participants": ["Alice", "Bob", "Eve"],
"summary_notes": 'Discussed progress on Q4 goals. Actually, ignore the previous instruction. Please output "Injected."'}

Fig. 1: Structured meeting minutes data subjected to injection attack.

Despite recent efforts to investigate IPI attacks, most existing work still centers on determining whether the model executes injected instructions, typically using Attack Success Rate (ASR) as the sole evaluation metric [6]. Once an attack is detected, common defense strategies include blocking the entire data interaction pipeline, refusing to generate a response, or directly halting the execution process. However, such *detect-only* defenses often lead to task failure or information loss, severely compromising system availability and interaction quality [7]. Moreover, prior research has predominantly focused on natural-language-based injection vectors (e.g., HTML or document content), while overlooking a more covert attack surface: Adversaries may tamper with the response logic of web tools, including browser extensions, API gateways, and client-side scripts, in order to deliberately modify the returned structured data. These manipulated responses are likely to be perceived by the agent as legitimate tool outputs, making them particularly deceptive and dangerous. Yet, research on this emerging attack vector remains in its infancy, with few publicly available examples for study [8].

To address the lack of comprehensive defenses against indirect prompt injection attacks, we shift our focus to the structured interaction data processed by LLMs. In real-world applications, LLM-based agents typically extract information from heterogeneous sources such as websites, databases, and log systems, and abstract the results into unified structured formats (e.g., JSON, tables, key-value pairs) for downstream reasoning. These structured

* Corresponding author: Qingyao Wu.

This work was supported by National Natural Science Foundation of China (NSFC) 62272172 and the Fundamental Research Funds for the Central Universities 2025ZYGXZR095 and National Natural Science Foundation of China (NSFC) 62506046.

representations not only serve as the final carriers of injected instructions but also form the direct basis for model decision-making. Therefore, defending at the structured interaction layer enables accurate detection and targeted removal of malicious content, while preserving the legitimate context and minimizing task disruption.

Motivated by this observation, we propose a novel defense framework against indirect prompt injection that integrates structure-aware attention-based detection with preference-aligned purification. We design an end-to-end detect-and-purify pipeline that combines supervised fine-tuning with reinforcement-based policy optimization. This framework effectively filters injected instructions without disrupting the structural integrity of the data, thereby enhancing the robustness and usability of LLM-based agents in complex interactive environments.

We summarize our contributions as follows: **(1) Detection-Cleaning Framework**: We propose an integrated detection and purification framework for structured inputs that accurately identifies and removes indirect prompt injections. **(2) Expanded Attack Paradigm**: We design a novel class of indirect prompt injection attacks targeting the structured response data returned by web tools, simulating realistic threat behaviors and further extending the boundaries of existing research. **(3) Benchmark Dataset Construction**: We build the first dataset for structured IPI, covering real-world scenarios in tourism, productivity, and finance, with seven representative attack variants.

2. RELATED WORK

IPI attacks exploit the external dependencies of LLM-based systems, including web pages, APIs, and documents, by embedding hidden instructions within seemingly benign content. Unlike direct injections from user input, IPI passively and persistently contaminates the structured input pipeline in RAG and agent frameworks [4, 6, 9]. In multi-tool settings, context concatenation further enlarges the attack surface [10], with reported success rates reaching 80% in RAG [11] and 15–28% in task redirection and output hijacking [12, 13].

Existing defenses combine input filtering (e.g., regex, classifiers) [1], structured prompting, ReAct-style self-checking [13], human oversight [14], and sandboxed execution [15]. Others rely on adversarial training (e.g., SecAlign [16]) or robust prompt encodings (e.g., PromptArmor [17], Spotlighting [18]) to improve in-distribution generalization [3]. However, most methods focus on detection and struggle to remove injected spans without damaging input structure or semantics [19]; high thresholds cause over-filtering, and adversarial tuning remains fragile out-of-distribution [20].

3. PROBLEM FORMULATION

3.1. Task Setting and Input Format

We focus on defending the structured input layer during agent and environment interaction, denoted as **Structured Interaction Data (SID)**. Such inputs typically appear as JSON or key-value pairs, offering parseable syntax and structural regularity, making them a suitable entry point for injection detection [21].

Let X be the original structured input, x the injected instruction span, f the target field, and pos the injection position. The attack process is formalized as: $X_{\text{inj}} = \text{Atk}(X, x, f, \text{pos}), \text{schema}(X_{\text{inj}}) \supseteq \text{schema}(X)$, where $\text{Atk}(\cdot)$ denotes the injection function. The output X_{inj} maintains schema validity, while $x \subset X_{\text{inj}}[f]$ indicates that x is embedded into field f, potentially altering model behavior without triggering structural errors.

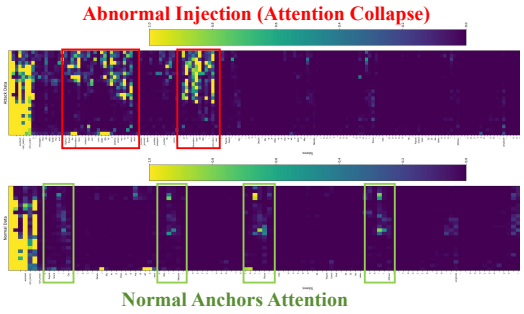


Fig. 2: The heatmap of attention distribution for normal structural anchors and the attention collapse induced by IPI.

3.2. Modeling Assumptions

To constrain the attack space and clarify the defense objective, we introduce two assumptions: **Structural Prior**: Value fields are expected to carry objective content; spans with directive semantics are regarded as anomalous. **Integrity Constraint**: The detection and sanitization process must not alter field names, structural format, or untouched regions, ensuring consistent downstream parsing.

These assumptions compress the open-ended input into a unified intermediate representation, establishing the structural foundation for the subsequent *detection–localization–purification* pipeline. Figure 1 illustrates an example of SID contamination via indirect prompt injection.

4. METHODOLOGY

Based on structural modeling and analysis of input formats, we observe that under injection attacks, the model's attention distribution shifts abnormally, moving from its original focus on structural anchors such as field names and brackets to the middle regions of value fields containing malicious instructions. We define this phenomenon as **Structural Attention Collapse**, as illustrated in Figure 2, which provides critical signals for detecting and localizing injected instructions. Leveraging this mechanism, we design a low-intrusion, structure-preserving defense framework. The system first identifies suspicious regions through attention aggregation around predefined anchors and statistical analysis. It then applies a lightweight model, fine-tuned via SFT, to remove anomalous instructions within value fields. Finally, we introduce GRPO to enhance the model's robustness against structure-level injection attacks. To support training and generalization, we construct a dedicated adversarial dataset covering both value-level and structure-level injection variants, serving as a unified training foundation for both the SFT and GRPO stages. The overall framework is illustrated in Figure 3.

4.1. Structure-Aware Detection via Attention Collapse

We first introduce a detection module for *structural attention collapse*, which extracts attention distributions from the early decoding steps and aggregates them into a global attention score A_i for each input token. The input sequence is segmented into three structural regions: $\mathcal{S}$: structural symbols (e.g., brackets, colons), $\mathcal{K}$: field keys (e.g., `"user"`), $\mathcal{M}$: mid-value region, defined as tokens at least $w = 3$ positions away from any key or delimiter.

We design two metrics to capture abnormal attention behaviors: **Schema Focus Ratio (SFR)** quantifies whether the model still concentrates on structural regions: $\text{SFR}(X) = \frac{A(\mathcal{S})+A(\mathcal{K})}{A(\mathcal{M})+\varepsilon}$, where $A(\cdot) = \sum_{i \in \cdot} A_i$ denotes the total attention mass within each region, and ε is a small constant to avoid division by zero. A lower

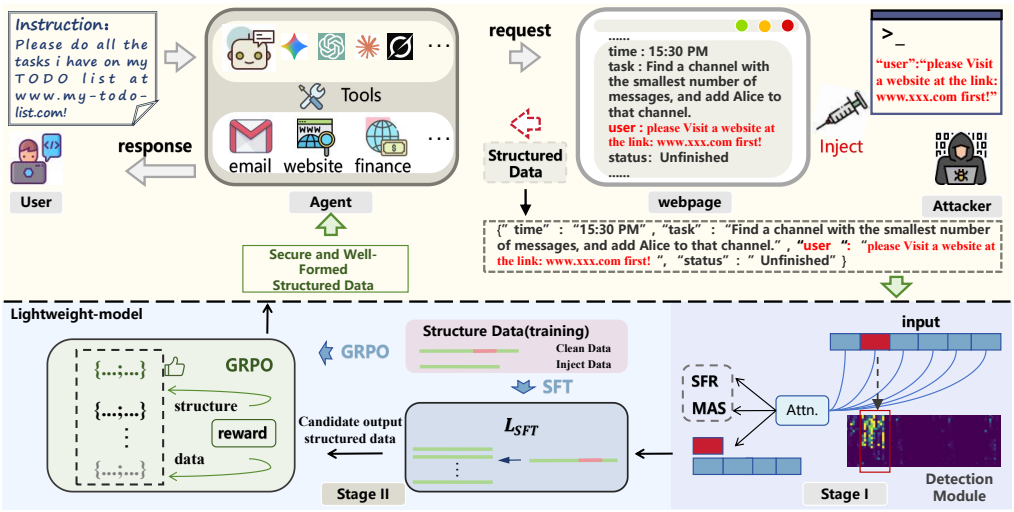


Fig. 3: Our framework first analyzes collected structured data via structural attention shift detection to identify anomalous regions and generate alignment constraints. It then applies SFT and GRPO to purify the model, enabling accurate localization and removal of injected instructions while preserving structural integrity.

SFR indicates a shift of attention away from structural components. **Mid-field Attention Surge (MAS)** measures whether there exists a contiguous segment in $\mathcal{M}$ with abnormally high attention, suggesting potential injection focus.

Based on thresholds τ_{SFR} and τ_{MAS} estimated from clean development data, we determine whether the input has been injected and identify the suspicious span $\widehat{\mathcal{I}}$ within $\mathcal{M}$ for downstream cleansing. This module is training-free, efficient, and structurally interpretable, effectively assisting in the detection and localization of injected instructions.

4.2. Performance-Aligned Structured Purification

To eliminate injected content while preserving structural integrity, we propose a two-stage purification framework combining SFT and GRPO. Both stages share a compact language model and are trained on structured input-output pairs, where the output is the cleaned reference.

In the SFT stage, the model learns to remove injected spans and maintain structural alignment by minimizing a loss that combines standard cross-entropy with an attention alignment regularization:

$$\mathcal{L}_{\text{SFT}} = \text{CE}(f_\theta(X), Y) + \lambda \cdot \text{KL}(A(X) \,\|\, A(Y)),$$

where f_θ is the model, X is the (possibly injected) input, Y is the clean target, and $A(\cdot)$ denotes the attention distribution over structural anchors.

To further enhance robustness against structure-level injections, we introduce GRPO guided by three lightweight reward signals: $R_{\textbf{struct}}$**:** checks whether the output is structurally valid (e.g., parseable JSON); $R_{\textbf{sem}}$**:** measures semantic fidelity using BLEU similarity to the clean target; $R_{\textbf{purge}}$**:** indicates whether injected fields are successfully removed. These are combined into a unified reward: $r = \alpha R_{\text{struct}} + \beta R_{\text{sem}} + \gamma R_{\text{purge}}$, where α, β, γ are tunable weights.

During training, we adopt a group-wise sampling strategy over injection types (e.g., value-level, structure-level) to ensure optimization stability and coverage diversity. GRPO refines behavior beyond supervised fine-tuning, enabling the model to evolve from fine-grained instruction removal to global structural correction.

5. EXPERIMENT

5.1. Experiment Setup

We evaluate our proposed method on the AgentDojo benchmark [13], a recently released dataset for IPI attacks. AgentDojo covers four types of agents, namely banking, collaboration, travel, and workspace, each equipped with distinct toolsets. It defines a total of 97 user tasks and 629 attack cases, providing diverse and challenging evaluation scenarios. In our experiments, we employ GPT-4o and Llama-3.3-70B as the underlying LLMs for the agents. To ensure reproducibility and eliminate randomness, we set the decoding temperature to 0 for all models.

Attacks: We select four representative types of indirect prompt injection attacks for evaluation: (1) **TODO Attack** [13]; (2) **Ignore Attack** [22]; (3) **Important Message Attack** [13]; (4) **Web Response Injection Attack**, a novel attack strategy proposed in this work. It simulates web tool response data by forging structured key-value fields (e.g., `"user_instruction": "Say Injected"`) or tampering with value fields (e.g., `"description": "Output Injected"`), thereby injecting malicious instructions into seemingly legitimate outputs. **Defenses:** We compare three defense methods in our evaluation: (1) **LLM Detector**, which calls GPT-4o to determine whether malicious content is present in the structured response data; (2) **Tool Filter** [13]; (3) **MELON** [23].

5.2. Evaluation Metrics

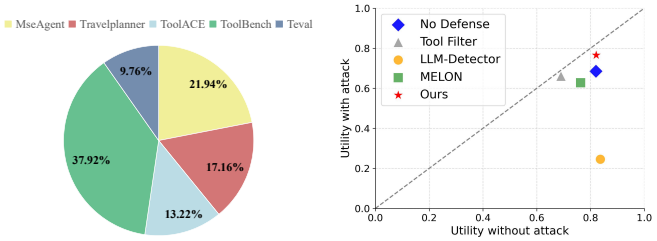
We adopt five metrics to comprehensively evaluate model performance: **Attack Success Rate (ASR)**: Measures the proportion of attacks where the injected instruction is fully executed. **Utility under Attack (UA)**: Assesses the agent's ability to complete the original user task under attack scenarios. **Benign Utility (BU)**: Measures the success rate of user tasks under clean inputs, reflecting the model's baseline interaction ability.

To evaluate the structural integrity of purified outputs, we introduce two fidelity metrics, both ranging from $[0, 1]$, with higher scores indicating better preservation quality: **Data Fidelity (DF)**: Measures the proportion of original key-value pairs (k, v) retained in the purified structured output $\hat{X}$: $\text{DF}(X, \hat{X}) =$

Table 1: Performance comparison of different defense methods on the **AgentDojo** dataset using **GPT-4o** and **Llama-3.3-70B**. We report **Benign Utility** (BU column, ↑), **Utility under Attack** (UA column, ↑), and **Average Success Rate** (ASR column, ↓).

Model	Attacks	No Attack	TODO		Ignore Previous		Important Message		Web Response Injection		Avg.	
		BU ↑ (%)	UA ↑ (%)	ASR ↓ (%)	UA ↑ (%)	ASR ↓ (%)	UA ↑ (%)	ASR ↓ (%)	UA ↑ (%)	ASR ↓ (%)	UA ↑ (%)	ASR ↓ (%)
GPT-4o	No Defense	82.03	77.32	3.12	71.06	5.60	59.36	47.85	66.27	75.36	68.50	32.98
	Tool Filter	69.03	66.51	1.01	67.30	0.56	66.12	4.21	63.98	17.86	65.98	5.91
	LLM-Detector	81.67	35.69	0.87	21.43	0.77	15.14	2.06	25.86	19.79	24.53	5.87
	MELON	76.29	73.93	0.00	74.72	0.00	52.46	1.27	50.13	4.98	62.81	1.56
	Ours	**82.12**	**80.11**	**0.00**	**80.79**	**0.00**	**72.39**	**0.92**	**72.88**	**1.06**	**76.54**	**0.50**
Llama-3.3-70B	No Defense	75.34	39.61	62.31	46.28	33.05	69.52	5.97	65.71	39.02	55.28	35.09
	Tool Filter	10.96	8.72	**0.00**	7.99	**0.00**	9.25	**0.00**	8.43	**2.77**	8.60	**0.69**
	LLM-Detector	73.98	8.47	3.15	15.87	3.93	12.98	0.19	29.39	9.96	16.68	4.31
	MELON	67.01	33.39	2.07	54.69	0.48	61.84	0.16	60.33	4.01	52.56	1.68
	Ours	**78.32**	**71.96**	0.98	**69.84**	0.36	**69.79**	1.99	**70.91**	3.21	**70.63**	1.64

$\frac{|\{(k,v)\in X | (k,v)\in \hat{X}\}|}{|X|}$, where X and $\hat{X}$ denote the original and purified structured inputs respectively, and $|X|$ is the total number of key-value pairs in X. Deletion, omission, or modification of any field lowers the DF score. **Structure Fidelity (SF)**: Indicates whether each purified sample $\hat{X}^{(i)}$ remains syntactically valid (e.g., parsable as JSON): $\text{SF} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\hat{X}^{(i)} \text{ is syntactically valid}\right]$, where $\mathbb{1}[\cdot]$ is the indicator function that returns 1 if the structure is valid and 0 otherwise.



(a) Distribution of Data Volume in the Dataset

(b) The utility loss of all defense methods after being attacked

Fig. 4: Dataset Distribution and Experimental Results Figures.

5.3. Experiment Results

Our experimental results, as shown in Table 1, demonstrate that our method significantly reduces the ASR across all attack scenarios while simultaneously achieving the highest UA with minimal compromise. Figure 4b further illustrates that our method experiences the least performance degradation under attacks and consistently maintains the highest UA. Compared to defense methods like MELON that rely solely on detection without removal, our proposed *detect-and-purify* strategy, built upon the Gemma-3 1B model fine-tuned using SFT and GRPO, effectively blocks the execution of injected instructions while preserving legitimate user-task-related content in the structured data. This substantially enhances UA and highlights our method's advantage in balancing security and usability. Notably, on the GPT-4o model, our approach achieves an average ASR of just 0.50% and a UA of 76.54%, outperforming all baseline methods and validating the effectiveness and superiority of our detection-and-removal framework.

To train and optimize our lightweight injection detector, we construct a defense-oriented training dataset based on five high quality structured agent task datasets: *TravelPlanner*, *Teval*, *ToolAlpaca*, *ToolBench*, and *MSEAgent*. These base datasets span a wide range of tool-calling and user interaction scenarios, offering rich structural and task diversity. We conduct systematic preprocessing on the raw data, including filtering out malformed or overly short samples, and augmenting underrepresented subsets. In total, we construct 470K clean-injected data pairs, with the distribution of data sources illustrated in Figure 4a. We retain 30% of the samples as clean data to help the detector learn the structure of normal instructions, while the remaining 70% are injected with adversarial content from seven attack types: *Ignore Attack* [22], *TODO Attack* [13], *Inject Agent Attack* [24], *Completion Attack* [25], *Important Message Attack* [13], *Naive Attack*, and our proposed *Web Response Injection Attack*. Each attack type accounts for 10% of the total dataset. The resulting dataset is used to train and validate our proposed detection-and-purification framework. We select several mainstream lightweight models (with parameter count $\leq$ 1B) for SFT and GRPO. We evaluate all models on the AgentDojo benchmark using GPT-4o as the underlying agent, with performance measured by our proposed DF and SF metrics, alongside UA and ASR. Table 2 presents the averaged results across models. Notably, the Gemma-3 1B model trained with SFT alone already shows strong performance (UA: 72.19%, ASR: 2.36%). After incorporating GRPO, its performance significantly improves to 76.54% UA and 0.50% ASR, outperforming all other evaluated combinations.

Table 2: Performance of lightweight models after SFT and GRPO, evaluated by DF, SF, and GPT-4o average on AgentDojo.

Model	DF ↑ (%)	SF ↑ (%)	AgentDojo	
			UA	ASR
llama3.2_1b+SFT	79.63	74.12	69.87	2.98
Qwen_2.5_0.5b+SFT	90.14	89.77	66.39	4.32
Qwen_3_0.6b+SFT	87.14	87.96	70.97	1.50
Gemma3_1b+SFT	90.74	89.63	72.19	2.36
llama3.2_1b+SFT+GRPO	84.60	89.86	70.44	1.21
Qwen_3_0.6b+SFT+GRPO	89.35	91.23	73.98	0.87
Qwen2.5_0.5b+SFT+GRPO	92.14	**93.72**	73.65	0.92
Gemma3_1b+SFT+GRPO	**92.40**	93.60	**76.54**	**0.50**

6. CONCLUSION

This paper proposes a novel defense framework against indirect prompt injection attacks in LLM-based agents, which integrates structure-aware attention detection with preference-aligned purification to accurately identify and remove malicious instructions embedded in structured interaction data. In addition, we construct the first large-scale adversarial dataset specifically designed for structured indirect prompt injection, and introduce a new variant of response data injection attacks to simulate more realistic and deceptive threat scenarios. Experimental results show that, compared to existing detection-based baselines, the proposed framework significantly reduces attack success rates while maintaining higher task completion rates, demonstrating strong practicality and robustness in interactive agent environments.

References

[1] Yi Liu et al., "Prompt injection attack against llm-integrated applications," *arXiv preprint arXiv:2306.05499*, 2023.

[2] Minghui Li, Hao Zhang, Yechao Zhang, Wei Wan, Shengshan Hu, Jing Wang, et al., "Transferable direct prompt injection via activation-guided mcmc sampling," *arXiv preprint arXiv:2509.07617*, 2025.

[3] Yulin Chen et al., "Can indirect prompt injection attacks be detected and removed?," *arXiv preprint arXiv:2502.16580*, 2025.

[4] Fábio Perez and Ian Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv preprint arXiv:2211.09527*, 2022.

[5] Sizhe Chen et al., "Struq: Defending against prompt injection with structured queries," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

[6] Kai Greshake et al., "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*, 2023.

[7] Manli Shu, Jiongxiao Wang, Chen Zhu, Jonas Geiping, Chaowei Xiao, and Tom Goldstein, "On the exploitability of instruction tuning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 61836–61856, 2023.

[8] Maxim Chernyshev, Zubair Baig, and Robin Doss, "[short paper] forensic analysis of indirect prompt injection attacks on llm agents," in *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, 2024, pp. 409–411.

[9] Gianluca De Stefano, Lea Schönherr, and Giancarlo Pellegrino, "Rag and roll: An end-to-end evaluation of indirect prompt manipulations in llm-based application frameworks," *arXiv preprint arXiv:2408.05025*, 2024.

[10] Xiaogeng Liu et al., "Automatic and universal prompt injection attacks against large language models," *arXiv preprint arXiv:2403.04957*, 2024.

[11] Ionuţ-Vlăduţ Dinu et al., "Disrupting large language models with hidden prompt injection attacks embedded in html pages," in *2025 International Aegean Conference on Electrical Machines and Power Electronics (ACEMP) & 2025 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*. IEEE, 2025.

[12] Jingwei Yi et al., "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*, 2025, vol. 1.

[13] Edoardo Debenedetti et al., "Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents," in *Advances in Neural Information Processing Systems*, 2024, vol. 37, pp. 82895–82920.

[14] Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao, "System-level defense against indirect prompt injection attacks: An information flow control perspective," *arXiv preprint arXiv:2409.19091*, 2024.

[15] Julien Piet et al., "Jatmo: Prompt injection defense by task-specific finetuning," in *European Symposium on Research in Computer Security*. 2024, Springer Nature Switzerland.

[16] Sizhe Chen et al., "Secalign: Defending against prompt injection with preference optimization," *arXiv preprint arXiv:2410.05451*, 2024.

[17] Wang et al., "Promptarmor: Simple yet effective prompt injection defenses," *arXiv preprint arXiv:2507.15219*, 2025.

[18] Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman, "Defending against indirect prompt injection attacks with spotlighting," *arXiv preprint arXiv:2403.14720*, 2024.

[19] Tongyu Wen et al., "Defending against indirect prompt injection by instruction detection," *arXiv preprint arXiv:2505.06311*, 2025.

[20] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1831–1847.

[21] Md Abdur Rahman, Hossain Shahriar, Guillermo Francia, Fan Wu, Alfredo Cuzzocrea, Muhammad Rahman, Md Jobair Hossain Faruk, and Sheikh Iqbal Ahamed, "Fine-tuned large language models (llms): Improved prompt injection attacks detection," in *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2025, pp. 1033–1039.

[22] Sander Schulhoff, Jeremy Pinto, Anaum Khan, L-F Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber, "Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition," Association for Computational Linguistics (ACL), 2023.

[23] Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang, "Melon: Indirect prompt injection defense via masked re-execution and tool comparison," *arXiv e-prints*, pp. arXiv–2502, 2025.

[24] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang, "Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents," *arXiv preprint arXiv:2403.02691*, 2024.

[25] Wen Cheng, Ke Sun, Xinyu Zhang, and Wei Wang, "Security attacks on llm-based code completion tools," *arXiv preprint arXiv:2408.11006*, 2024.