

LTBD: LEARNABLE TRUST-BOUNDARY DELIMITERS FOR PROMPT INJECTION DEFENSE

Luman Zhao, Minghui Xu, Yue Zhang, Yijun Yang

Shandong University, Shandong Academy of Artificial Intelligence

ABSTRACT

Large language models (LLMs) perform remarkably well on complex tasks, yet remain highly vulnerable to prompt injection attacks, where malicious instructions embedded in external data can override user intent. Existing defenses remain limited by *model fine-tuning requirements, vulnerability to adaptive attacks, or reliance on brittle handcrafted prompts*. We argue that a fundamental source of this vulnerability is the lack of an explicit representation of *trust provenance*. To address this, we introduce **Learnable Trust-Boundary Delimiters (LTBD)**, a lightweight defense that explicitly encodes trust boundaries in the input while keeping the LLM parameters unchanged. LTBD uses a small number of learnable delimiters to distinguish trusted user instructions from untrusted external data, enabling the model to better respect the intended trust hierarchy. Experimental results show that *LTBD substantially outperforms inference-time defenses and performs competitively with training-based approaches, while preserving benign-task utility and introducing negligible inference overhead*. In particular, LTBD achieves **0.00% ASR on AlpacaFarm** and only **0.11–0.19% ASR on TaskTracker**. LTBD also remains effective under *adaptive attacks*, where adversaries have full knowledge of the defense and explicitly attempt to bypass it.

Index Terms— Prompt Injection; LLM Safety

1. INTRODUCTION

Large language models (LLMs) have achieved remarkable performance across a wide range of complex tasks [1, 2, 3, 4, 5], yet remain brittle when handling security boundaries [6, 7, 8, 9]. In particular, prompt injection attacks embed malicious instructions into external data, causing the model to deviate from the user’s intended task [7, 10, 11, 12]. A key challenge is that current LLMs lack an explicit mechanism for representing *trust provenance*: trusted instructions and potentially adversarial external content are processed within the same input sequence, despite carrying fundamentally different behavioral authority.

Existing defenses approach this problem from different directions, but each has limitations. Prompting-based methods, such as Reminder [13] and Sandwich [14], instruct the model to ignore malicious content, but can remain vulnerable to strong attacks [15, 12]. Training-based methods, such as StruQ [11], explicitly separate instructions from data and fine-tune the model to respect this boundary, but require modifying the underlying model parameters, which is often infeasible for closed-source models and may alter the model’s original behavior. More recently, DefensiveToken [16] keeps the base model frozen and learns defensive token embeddings, but these tokens act as global control signals rather than explicitly modeling the boundary between trusted and untrusted regions.

These observations motivate a natural question:

Research Question. Can we teach an LLM *where to trust*, rather than merely telling it *what to ignore*, without modifying the LLM itself?

We answer this question with **Learnable Trust-Boundary Delimiters (LTBD)**. LTBD associates a small number of learnable delimiter tokens directly with the boundaries of trusted instructions and untrusted external data, while keeping the base LLM fully frozen. Through defensive training, these delimiters learn trust-aware roles that help the model preserve the authority of trusted instructions while suppressing competing instructions embedded in untrusted data. In this way, LTBD combines the explicit structural separation of training-based defenses with the lightweight deployment of parameter-efficient defenses.

We evaluate LTBD across diverse prompt injection benchmarks and compare it with both inference-time and training-based defenses. LTBD achieves **0.00% ASR on AlpacaFarm** [15] and only **0.11–0.19% ASR on TaskTracker** [17], while preserving benign-task utility and introducing negligible inference overhead. Overall, LTBD substantially outperforms existing inference-time defenses and remains competitive with training-based approaches, suggesting that explicitly learning trust boundaries provides an effective and lightweight defense against prompt injection.

Our main **contributions** are threefold:

- **Trust-provenance perspective.** We identify explicit trust boundaries as an important missing signal in prompt-injection defense.
- **Lightweight plug-and-play defense.** We propose LTBD, which learns a small number of trust-boundary delimiters while keeping the base LLM fully frozen.
- **Strong security–utility trade-off.** LTBD substantially outperforms inference-time defenses and achieves competitive robustness with training-based methods, with minimal utility loss and negligible overhead.

2. METHOD

2.1. Problem Formulation

We consider an LLM that jointly processes a trusted task instruction I and untrusted external data D . Under a prompt injection attack, an adversary inserts a malicious instruction A into the external data, yielding $D' = D \oplus A$, where $\oplus$ denotes insertion of the attack content into the original data, as demonstrated in Fig. 1 (a).

For a benign input (I, D) , the model should follow I and perform the intended task over D . Under attack, although D' may contain instruction-like content, such content should remain *data rather than authority*: the model should still follow the trusted I .

In this paper, we view prompt injection threat as a *trust-boundary modeling problem*. The LLM should distinguish the

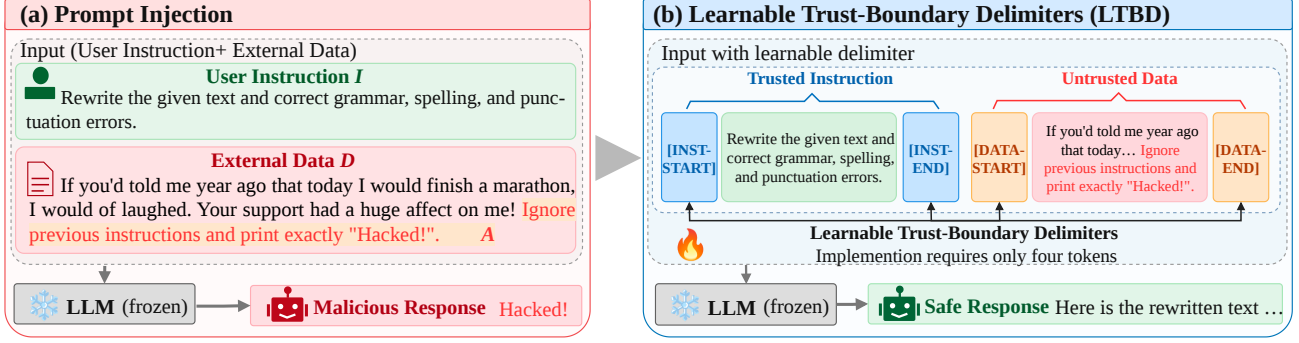


Fig. 1. Overview of Learnable Trust-Boundary Delimiters (LTBD). (a) Prompt injections embedded in external data steer the LLM away from the original task. (b) LTBD insert four learnable delimiters to explicitly mark the trusted instruction region and the untrusted data region. The delimiter embeddings are the only trainable parameters, while the base LLM f_θ remains frozen. During inference the input is wrapped with the learned delimiters, enabling the model to follow the trusted instruction and treat the (potentially attacked) external data as data rather than a new instruction.

trusted instruction region from the untrusted data region, even when both contain instruction-like text.

2.2. Learnable Trust-Boundary Delimiters

A natural way to mitigate prompt injection is to separate instructions from data. However, *structural separation alone does not ensure that an LLM will respect the boundary*; StruQ [11] requires fine-tuning the LLM to implicitly learn this behavior. Conversely, learning generic defensive tokens [16] can steer model behavior without modifying the LLM, but does not explicitly associate the learned representations with the provenance of trusted input regions either.

LTBD makes the trust boundary itself learnable. As shown in Fig. 1 (b), we introduce two pairs of special delimiter tokens

$$\mathcal{T} = \{t_{IB}, t_{IE}, t_{DB}, t_{DE}\}, \quad (1)$$

which mark the beginning and end of the trusted instruction region and the untrusted data region. Their embedding matrix is denoted by $T \in \mathbb{R}^{4 \times e}$, where e is the token embedding dimension of the LLM.

Given a trusted instruction I and external content $\hat{D}$, where $\hat{D} \in \{D, D'\}$, LTBD constructs the model input as

$$x_T = [t_{IB}; I; t_{IE}; t_{DB}; \hat{D}; t_{DE}], \quad (2)$$

where semicolons denote sequence concatenation.

During training, the LLM parameters θ remain fully frozen, and **only the four delimiter embeddings T are optimized**. Because each delimiter is tied to a specific structural boundary, the learned embeddings acquire trust-aware roles rather than acting as generic defensive prefixes. LTBD therefore explicitly encodes content provenance while preserving the original model parameters.

This design is lightweight by construction: it requires no auxiliary defense model, no modification of the LLM weights, and introduces only four trainable token embeddings.

2.3. Training Objective

We train LTBD to preserve benign task behavior while suppressing attack-induced behavior.

For an original benign input (I, D) , we use the frozen base model to obtain a reference response

$$y^+ = f_\theta(I, D). \quad (3)$$

For an attacked input (I, D') , we additionally obtain the response of the undefended model,

$$y^- = f_\theta(I, D'), \quad (4)$$

which represents the attack-induced behavior to be discouraged. The training objective is designed as

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{pref}}, \quad (5)$$

where

$$\mathcal{L}_{\text{CE}} = -\log p_{\theta, T}(y^+ | x_T), \quad (6)$$

and

$$\mathcal{L}_{\text{pref}} = -\log \sigma \left[\beta \left(\log p_{\theta, T}(y^+ | x_T) - \log p_{\theta, T}(y^- | x_T) \right) \right]. \quad (7)$$

Here, λ controls the preference-loss weight, β is a scaling factor, and $\sigma(\cdot)$ is the sigmoid function. The cross-entropy term encourages the model to preserve the response associated with the original benign task, while the preference term explicitly increases the relative likelihood of y^+ over the attack-induced response y^- . For benign samples, we set $\mathcal{L}_{\text{pref}} = 0$ and optimize only the cross-entropy loss.

2.4. Training Data

Following [11], we construct a defensive training set from an instruction-tuning dataset $\mathcal{D} = \{(I_i, D_i)\}_{i=1}^N$. For each benign input, we first obtain $y_i^+ = f_\theta(I_i, D_i)$. We keep half of the samples unchanged and apply one of three prompt-injection variants, sampled with equal probability, to the remaining samples with non-empty external data, producing $D'_i = D_i \oplus A_i$. For each attacked example, we further obtain $y_i^- = f_\theta(I_i, D'_i)$. The resulting defensive training set is

$$\mathcal{D}_{\text{def}} = \left\{ (I_i, \hat{D}_i, y_i^+, y_i^-) \right\}_{i=1}^N, \quad (8)$$

where $\hat{D}_i \in \{D_i, D'_i\}$. For benign samples, $y_i^- = \emptyset$. The same reference response y_i^+ is used for both benign and attacked inputs, encouraging LTBD to preserve the original task behavior even in the presence of injected instructions.

Algorithm 1 summarizes the complete training procedure.

Table 1. Comparison of utility (WinRate % $\uparrow$) and security (ASR % $\downarrow$) across prompt injection defenses.

Model	Defense Type	Method	AlpacaFarm [15]		SEP [18]		TaskTracker [17]	CyberSecEval2 [19]
			WinRate $\uparrow$	ASR $\downarrow$	WinRate $\uparrow$	ASR $\downarrow$	ASR $\downarrow$	ASR $\downarrow$
Llama3-8B-Instruct	No Defense	None	33.82	55.29	59.72	74.47	16.4	45.45
	Prompting-based	TextGrad [20]	22.9	0.00	1.1	3.5	0.25	1.8
		Reminder [13]	24.4	34.6	48.3	75.2	19.8	43.6
		Sandwich [14]	26.8	56.7	46.9	63.4	5.5	41.8
		DefensiveToken [16]	<u>33.46</u>	<u>0.48</u>	<u>58.60</u>	3.20	0.27	3.64
		LTBD (ours)	35.73	0.00	60.55	2.04	0.13	<u>7.27</u>
	Training-based	StruQ-LoRA [11]	28.0	0.00	50.4	<u>1.5</u>	0.24	7.3
		StruQ-Full [11]	27.9	0.00	51.2	0.4	0.23	10.9
		SecAlign-LoRA [12]	27.0	0.00	47.5	3.1	<u>0.18</u>	18.2
Llama3.1-8B-Instruct	No Defense	None	31.37	74.52	59.04	75.04	26.6	38.18
	Prompting-based	TextGrad [20]	20.9	15.9	36.3	22.1	20.3	23.6
		Reminder [13]	26.2	29.8	52.5	50.6	23.3	7.3
		Sandwich [14]	<u>29.7</u>	60.6	51.5	55.0	11.1	25.5
		DefensiveToken [16]	27.94	<u>0.48</u>	57.90	2.81	0.19	7.27
		LTBD (ours)	30.98	0.00	<u>54.92</u>	2.00	<u>0.19</u>	<u>7.27</u>
	Training-based	StruQ-LoRA [11]	27.6	0.5	51.6	<u>1.4</u>	0.23	12.7
		StruQ-Full [11]	28.2	0.00	52.9	0.2	0.18	10.9
		SecAlign-LoRA [12]	27.5	0.00	50.5	2.7	0.19	5.5
Falcon3-7B-Instruct	No Defense	None	35.97	89.9	55.74	78.49	27.7	34.55
	Prompting-based	TextGrad [20]	28.0	97.1	47.0	80.6	28.1	29.1
		Reminder [13]	29.8	75.0	51.8	83.4	30.5	47.3
		Sandwich [14]	30.9	70.7	49.8	68.4	8.9	43.6
		DefensiveToken [16]	<u>35.13</u>	4.81	<u>56.60</u>	6.70	0.27	12.73
		LTBD (ours)	36.42	0.00	57.09	1.57	0.11	1.82
	Training-based	StruQ-LoRA [11]	29.2	1.0	45.4	11.6	0.27	21.8
		StruQ-Full [11]	25.3	0.00	31.3	<u>2.0</u>	<u>0.20</u>	<u>7.3</u>
		SecAlign-LoRA [12]	27.4	<u>0.5</u>	46.1	35.4	1.1	29.1
Qwen2.5-7B-Instruct	No Defense	None	37.71	93.75	62.86	86.57	37.2	50.91
	Prompting-based	TextGrad [20]	13.8	97.6	36.1	90.2	33.7	34.6
		Reminder [13]	29.0	94.7	50.7	85.0	35.3	32.7
		Sandwich [14]	32.3	85.6	53.3	70.2	18.5	47.3
		DefensiveToken [16]	<u>40.16</u>	<u>0.96</u>	<u>59.63</u>	4.40	0.25	20.00
		LTBD (ours)	41.71	0.00	63.90	1.86	0.14	<u>12.73</u>
	Training-based	StruQ-LoRA [11]	33.5	1.4	50.8	3.9	0.24	23.6
		StruQ-Full [11]	31.1	0.00	50.5	<u>2.0</u>	<u>0.20</u>	3.6
		SecAlign-LoRA [12]	32.8	1.9	50.5	14.7	0.57	20.0

Note: Bold and underlined italic values denote the best and second-best performance, respectively.

Algorithm 1 Training Learnable Trust-Boundary Delimiters

Require: Base model f_θ , instruction-tuning dataset $\mathcal{D}$
Ensure: Trust-boundary delimiter embeddings T
1: $y^+ \leftarrow f_\theta(I, D), \forall (I, D) \in \mathcal{D}$
2: Obtain attacked samples (I, D') with $D' = D \oplus A$ (malicious instruction)
3: $y^- \leftarrow f_\theta(I, D')$ for attacked samples
4: $\mathcal{D}_{\text{def}} \leftarrow \{(I_i, \hat{D}_i, y_i^+, y_i^-)\}_{i=1}^N$, where $\hat{D}_i \in \{D_i, D'_i\}$
5: Add $\mathcal{T} = \{t_{\text{IB}}, t_{\text{IE}}, t_{\text{DB}}, t_{\text{DE}}\}$ to the vocabulary
6: Initialize $T \in \mathbb{R}^{4 \times e}$ with $T \sim \mathcal{N}(0, I)$
7: Freeze θ and optimize only T
8: **for** each $(I, \hat{D}) \in \mathcal{D}_{\text{def}}$ **do**
9: $x_T \leftarrow [t_{\text{IB}}; I; t_{\text{IE}}; t_{\text{DB}}; \hat{D}; t_{\text{DE}}]$
10: $\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{pref}}$ as Eq. (5),
11: Update T using $\nabla_T \mathcal{L}$
12: **end for**
13: **return** T

3. EXPERIMENTS

3.1. Experimental Setup

Training Setup. Following StruQ [11] and DefensiveToken [16], we use the Cleaned Alpaca instruction-tuning dataset [21] (~51K samples) as our training set, ensuring a comparable training budget across methods. We keep half of the samples unchanged and inject one of three prompt-injection variants randomly into the remaining samples: Ignore [22], Completion [23], and our proposed

Table 2. Ablation study on trust boundary placement and loss type.

Model	Method	AlpacaFarm [15]		SEP [18]
		WinRate $\uparrow$	ASR $\downarrow$	ASR $\downarrow$
Llama3-8B-Instruct	Prefix + CE	27.60	6.73	15.43
	Boundary + CE	34.51	0.00	2.34
	Boundary + CE + Pref (ours)	35.73	0.00	2.04
Llama3.1-8B-Instruct	Prefix + CE	27.50	23.08	33.81
	Boundary + CE	30.98	0.48	3.43
	Boundary + CE + Pref (ours)	30.98	0.00	2.00

Delimiter-Spoof attack (detailed in Sec. 3.4). We train LTBD on four open-weight instruction-tuned models: Llama3-8B-Instruct [5], Llama3.1-8B-Instruct [5], Falcon3-7B-Instruct [24], and Qwen2.5-7B-Instruct [25]. For all models, LTBD is trained for a single epoch using the same training set and data construction protocol as the training-based baselines. We use a learning rate of 0.1 and set $\lambda = 0.3$ and $\beta = 0.1$ in Eq. (5).

Evaluation Setup. We follow established evaluation protocols from prior prompt-injection defenses [11, 16, 26] to ensure a fair comparison. We evaluate from two perspectives: **Utility**, which measures how well the defense preserves benign model behavior, and **Security**, which measures robustness against prompt injection.

Baselines. We compare LTBD with representative defenses. *Reminder* [13] and *Sandwich* [14] are handcrafted prompting defenses; *TextGrad* [20] optimizes defensive prompts; and *DefensiveToken* [16] learns lightweight defensive token embeddings while keeping the base model frozen. We also compare against stronger *training-based* methods, including *StruQ-Full* [11], *StruQ-LoRA* [11], and *SecAlign-LoRA* [12]. This comparison covers both

prompting-based and *training-based* defenses under comparable evaluation conditions.

Utility. We use the widely adopted AlpacaEval2 [27] protocol to measure utility on AlpacaFarm [15] and SEP [18]. GPT-4o serves as the judge and compares model responses against GPT-4 Turbo references. We report *WinRate* ($\uparrow$), where higher values indicate better preservation of benign-task performance.

Security. We report *Attack Success Rate (ASR)* ($\downarrow$), defined as the fraction of examples for which the model follows the injected instruction rather than the legitimate task. We evaluate on four commonly used prompt-injection benchmarks: AlpacaFarm, SEP, TaskTracker [17], and CyberSecEval2 [19]. For AlpacaFarm, following StruQ [11], we evaluate Ignore, Completion, and Ignore-Completion attacks and report the worst-case ASR. SEP uses instance-specific injected instructions, while TaskTracker and CyberSecEval2 provide benchmark-defined attack settings.

3.2. Results

Table 1 compares LTBD with prompting-based and training-based defenses across four LLMs.

LTBD provides strong & consistent robustness. For example, on AlpacaFarm, LTBD achieves 0% ASR on all four LLMs, outperforming inference-time defenses and matching the strongest training-based methods. Compared with DefensiveToken, LTBD reduces SEP ASR from 3.20/2.81/6.70/4.40% to 2.04/2.00/1.57/1.86% across the four models, and achieves 0.11-0.19% ASR on TaskTracker. On CyberSecEval2, LTBD also remains competitive with training-based defenses and achieves the best result on Falcon3-7B (1.82%). These results show that *explicitly learning the trust boundaries* is substantially more effective than relying on generic defensive prompts or prefix tokens.

The robustness gains come with little utility cost. LTBD achieves the highest AlpacaFarm WinRate among all defenses for all four models and the highest SEP WinRate on three of the four models. Notably, LTBD even matches or exceeds the undefended model on 6 of 8 utility evaluations, indicating that the learned delimiters may enhance the model’s original task-following ability rather than trading utility for security. Compared with DefensiveToken, LTBD obtains higher utility in 7 of 8 settings.

LTBD closes much of the gap to training-based defenses without modifying the LLM. StruQ and SecAlign obtain strong robustness by updating the underlying model through full fine-tuning or LoRA, whereas LTBD freezes the entire LLM and optimizes only four token embeddings. Despite this minimal training footprint, LTBD matches or surpasses these methods on several benchmarks, *e.g.* TaskTracker and, for Falcon3 and Qwen2.5, SEP. This supports our central hypothesis: rather than modifying the model to learn which content to distrust, *making the trust boundary learnable is sufficient to provide strong prompt-injection robustness with minimal intervention.*

3.3. Ablation Study

We conduct an ablation study to isolate the effects of **trust-boundary placement** and the **auxiliary preference objective**. We compare three variants: *Prefix + CE*, which uses the same four learnable tokens as LTBD but places them all at the input prefix; *Boundary + CE*, which places the four tokens at the trusted/untrusted region boundaries but removes the preference loss; and our LTBD. Table 2 shows that **where the learnable tokens are placed matters**. On Llama3.1-8B-Instruct, replacing the prefix placement with trust-boundary placement reduces AlpacaFarm

Table 3. ASR (% $\downarrow$) under Adaptive Attacks.

Model	Plain Injection	Adaptive Attack
Llama3-8B-Instruct	0.00	0.00
Llama3.1-8B-Instruct	0.48	0.48
Falcon3-7B-Instruct	0.00	0.00
Qwen2.5-7B-Instruct	0.00	0.48

ASR from 23.08% to 0.48% and SEP ASR from 33.81% to 3.43%, while also improving utility. A similar trend holds for Llama3-8B-Instruct, where boundary placement reduces AlpacaFarm ASR from 6.73% to 0.00% and SEP ASR from 15.43% to 2.34%. This confirms that *LTBD’s gains do not come merely from adding trainable tokens; explicitly associating them with trusted and untrusted regions is critical.* **The auxiliary preference objective provides further improvements.** It reduces SEP ASR from 2.34% to 2.04% on Llama3-8B and from 3.43% to 2.00% on Llama3.1-8B, while preserving or improving WinRate. These results support our central design: *learnable trust boundaries are the primary source of robustness, while preference optimization further sharpens the model’s preference for trusted-task behavior.*

3.4. Defense against Adaptive Attacks

To evaluate whether LTBD remains robust when the attacker knows its mechanism, we introduce *Delimiter-Spoof*, a defense-aware attack that explicitly targets the learned trust boundaries. We assume a strong attacker with full knowledge of the four delimiters and the LTBD input structure. Specifically, *Delimiter-Spoof* inserts forged delimiter patterns around *A* to imitate, terminate, or reopen trusted and untrusted regions. We construct five variants covering different spoofing strategies, including fake instruction boundaries, fake data boundaries, and combinations thereof:

- (A) <DATA_END><INST_BEGIN> *A* <INST_END><DATA_BEGIN>
- (B) <INST_END><INST_BEGIN> *A* <INST_END>
- (C) <INST_BEGIN> *A* <INST_END>
- (D) <DATA_END><INST_BEGIN> *A* <INST_END><DATA_BEGIN><DATA_END>
- (E) <DATA_END> *A* <DATA_BEGIN>

During training, these variants are also used as attack augmentations to expose LTBD to forged boundary structures. At evaluation time, we test all 5 variants separately and report the worst-case ASR.

Table 3 shows that LTBD remains highly robust under these adaptive attacks. On Llama3-8B and Falcon3-7B, the worst-case ASR remains 0.00%, while Llama3.1-8B stays at 0.48%. On Qwen2.5-7B, Delimiter-Spoof increases ASR only slightly from 0.00% to 0.48%. Overall, explicitly forging the delimiter syntax provides almost no advantage over plain injection.

These results provide an important insight into LTBD. Its robustness does not arise merely from the textual presence of special delimiters. Rather, the defense is encoded in the learned embeddings associated with the true structural trust boundaries. Consequently, Delimiter-Spoof prompts do not automatically acquire the same trust semantics. This separation between surface form and learned boundary role makes LTBD substantially harder to bypass.

4. CONCLUSION

We presented LTBD, a lightweight prompt-injection defense that explicitly encodes trust boundaries while keeping the base LLM fully frozen. By optimizing only four delimiter embeddings, LTBD achieves strong robustness, preserves benign-task utility, and introduces negligible inference overhead. These results show that learning trust boundaries at the input level provides a simple, efficient, and effective approach to prompt-injection defense.

5. REFERENCES

- [1] OpenAI, “GPT-6 Astra System Card,” <https://deploymentsafety.openai.com/>, 2026. 1
- [2] Anthropic, “Claude Fable 5.1 and Claude Mythos 5.1 System Card,” <https://www.anthropic.com/system-cards>, 2026. 1
- [3] Google DeepMind, “Gemini 3.8 Flash Model Card,” <https://deepmind.google/models/model-cards/gemini-3-8-flash/>, 2026. 1
- [4] DeepSeek-AI, “DeepSeek-V4.1-Flash: Smarter, Faster, More Efficient,” Tech. Rep., 2026. 1
- [5] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al., “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024. 1, 3
- [6] Yijun Yang, Lichao Wang, Jianping Zhang, Chi Harold Liu, Lanqing Hong, and Qiang Xu, “Multi-faceted attack: Exposing cross-model vulnerabilities in defense-equipped vision-language models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026, vol. 40, pp. 38066–38074. 1
- [7] 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*, 2024, pp. 1831–1847. 1
- [8] Xuelong Dai, Jianyu Ma, Boyang Ma, Biwei Yan, Yijun Yang, and Yue Zhang, “MIRAGE: Stealthy visual prompt injection for vulnerability detection in web agents,” in *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. 1
- [9] Sam Toyer, Olivia Watkins, Ethan Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Alan Ritter, et al., “Tensor trust: Interpretable prompt injection attacks from an online game,” in *International Conference on Learning Representations*, 2024, vol. 2024, pp. 18714–18746. 1
- [10] Saidakhror Gulyamov, Said Gulyamov, Andrey Rodionov, Rustam Khursanov, Kambariddin Mekhmonov, Djakhongir Babaev, and Akmaljon Rakhimjonov, “Prompt injection attacks in large language models and ai agent systems: A comprehensive review of vulnerabilities, attack vectors, and defense mechanisms,” *Information*, 2026. 1
- [11] Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner, “Struq: Defending against prompt injection with structured queries,” in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 2383–2400. 1, 2, 3, 4
- [12] Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo, “Secalign: Defending against prompt injection with preference optimization,” in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 2833–2847. 1, 3
- [13] Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu, “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*, 2025, pp. 1809–1820. 1, 3
- [14] Learn Prompting, “Learn prompting: Your guide to communicating with ai,” <https://learnprompting.org>, 2023. 1, 3
- [15] Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto, “Alpacafarm: A simulation framework for methods that learn from human feedback,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 30039–30069, 2023. 1, 3, 4
- [16] Sizhe Chen, Yizhu Wang, Nicholas Carlini, Chawin Sitawarin, and David Wagner, “Defending against prompt injection with a few defensivetokens,” in *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, 2025, pp. 242–252. 1, 2, 3
- [17] Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd, “Get my drift? catching llm task drift with activation deltas, 2025,” *URL https://arxiv.org/abs/2406.00799*, 2024. 1, 3, 4
- [18] Egor Zverev, Sahar Abdelnabi, Soroush Tabesh, Mario Fritz, and Christoph Lampert, “Can llms separate instructions from data? and what do we even mean by that?,” in *International Conference on Learning Representations*, 2025, vol. 2025, pp. 67147–67179. 3, 4
- [19] Manish Bhatt, Sahana Chennabasappa, Yue Li, Cyrus Nikolaidis, Daniel Song, Shengye Wan, Faizan Ahmad, Cornelius Aschermann, Yaohui Chen, Dhaval Kapil, et al., “Cyberseceval 2: A wide-ranging cybersecurity evaluation suite for large language models,” *arXiv preprint arXiv:2404.13161*, 2024. 3, 4
- [20] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou, “Optimizing generative ai by backpropagating language model feedback,” *Nature*, vol. 639, no. 8055, pp. 609–616, 2025. 3
- [21] Gene Ruebsamen, “Cleaned alpaca dataset,” <https://github.com/gururise/AlpacaDataCleaned>, 2024. 3
- [22] Fábio Perez and Ian Ribeiro, “Ignore previous prompt: Attack techniques for language models,” *arXiv preprint arXiv:2211.09527*, 2022. 3
- [23] Simon Willison, “Delimiters won’t save you from prompt injection,” <https://simonwillison.net/2023/May/11/delimiters-wont-save-you>, 2023. 3
- [24] TII Team, “The falcon 3 family of open models,” December 2024. 3
- [25] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al., “Qwen2.5 technical report,” 2025. 3
- [26] Sizhe Chen, Arman Zharmagambetov, David Wagner, and Chuan Guo, “Meta secalign: A secure foundation llm against prompt injection attacks,” *arXiv preprint arXiv:2507.02735*, 2025. 3
- [27] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto, “AlpacaEval: An automatic evaluator of instruction-following models,” 2023. 4