Assessing Defenses Against Prompt Injection Attacks in LLM Systems

Fadime Zeliha Seyhan
R&D Depratment
Beyond Guard
Istanbul, Türkiye
zeliha.seyhan@beyondguard.ai

Katira Soleymanzadeh
R&D Depratment
Beyond Guard
Istanbul, Türkiye
katira.soleymanzadeh@beyondguard.ai

Mustafa Oğuz Cengiz
R&D Depratment
Beyond Guard
Istanbul, Türkiye
mustafa.cengiz@beyondguard.ai

Alp Ecevit
R&D Depratment
Beyond Guard
Istanbul, Türkiye
alp.ecevit@beyondguard.ai

Roya Arkhmammadova
R&D Depratment
Beyond Guard
Istanbul, Türkiye
roya.arkhmammadova@beyondguard.ai

Elif Tansu Kaya
R&D Depratment
Beyond Guard
Istanbul, Türkiye
elif.kaya@beyondguard.ai

***Abstract*—Prompt injection poses a critical security and compliance risk for large language model (LLM)-integrated systems. This paper treats prompt injection as a system-level governance challenge and introduces a model-agnostic guardrail architecture that performs prompt-level risk assessment prior to LLM inference. As a core component, we develop a dedicated prompt injection detector trained using Adaptive Layer LoRA on an in-house annotated dataset. Experiments across Gemma and Qwen model families show that baseline prompted LLMs exhibit unstable precision-recall trade-offs and elevated false positive rates, while increasing prompt complexity alone does not improve robustness. In contrast, LoRA fine-tuned models achieve deployment-grade performance, reaching up to 99.05% accuracy with substantially reduced false positives, false negatives, and attack success rates.**

Keywords— Prompt Injection, Guardrail Systems, LLM Safety

I. INTRODUCTION

Large language models (LLMs) [1][2][3] have become foundational components of modern AI systems, enabling strong performance across tasks such as text generation, summarization, and information retrieval[4][5]. In real-world deployments, however, LLMs are rarely used in isolation. Instead, they are embedded within LLM-integrated applications that mediate interactions between users and backend models by combining system instructions with user- or externally supplied content into a single prompt. While this prompt-driven architecture enables flexible and expressive applications, it also introduces new security risks that arise directly from how instructions and data are jointly interpreted. These risks are increasingly relevant from a regulatory perspective, as emerging frameworks such as the EU AI Act [6][7] require deployed LLM systems to demonstrate robustness against adversarial manipulation and unintended behavior.

Among these risks, prompt injection has emerged as one of the most critical vulnerabilities affecting LLM-integrated systems [8][9][10]. Prompt injection attacks exploit the lack of strict separation between instructions and data, allowing adversarial inputs to override system prompts, bypass safety constraints, or redirect model behavior. Through carefully crafted inputs, attackers can induce LLMs to follow malicious objectives rather than the intended application logic, leading to outcomes such as policy circumvention, sensitive data exposure, or harmful content generation. As LLM-based systems are increasingly deployed in high-stakes domains, these failures pose not only alignment concerns but also broader risks to system reliability and trustworthiness. Reflecting their practical impact, prompt injection attacks are explicitly identified as a high-risk vulnerability in the OWASP Top 10 for LLM Applications [11].

Together, these security and regulatory considerations indicate that prompt injection is not merely a model-level weakness but a system-level risk that requires enforceable safeguards [12]. Prior work has shown that prompt injection attacks span a wide spectrum, from simple instruction-override patterns to more sophisticated optimization-based strategies that systematically steer model outputs toward adversarial goals [13][14]. In response, existing defenses have focused on prompt engineering, fine-tuning LLMs [15][16] for improved robustness, or applying post-hoc filtering and moderation. Although these approaches have shown partial success, empirical evidence suggests that many remain fragile under distribution shift, unseen attack patterns, or deployment-scale constraints, particularly when adversarial inputs are encountered at runtime.

This paper argues that prompt injection should be addressed as an architectural security problem rather than solely as a model alignment challenge. Instead of assuming that backend LLMs can reliably defend themselves against adversarial prompts, we advocate for an explicit guardrail layer operating at the input boundary of LLM-integrated systems. Such a layer enables systematic inspection and control of prompts before they reach task-specific models, providing protection that is independent of the backend model's internal training or alignment mechanisms.

To this end, we propose a model-agnostic, governance-oriented guardrail architecture that functions as a mandatory intermediary between users and backend LLM services. The guardrail performs real-time prompt-level risk assessment to detect potential prompt injection attempts prior to task execution and can be deployed consistently across heterogeneous models and providers without requiring modifications to the underlying LLMs. As a core component of this architecture, we design and evaluate a dedicated prompt injection detection model trained using an Adaptive Layer LoRA strategy [17], enabling efficient and targeted adaptation without full-parameter fine-tuning. The detector is trained on a large in-house dataset of annotated prompts covering diverse attack patterns and severity levels, with structured supervision to improve robustness.

We conduct extensive experiments across multiple model families and scales, including baseline LLMs, LoRA fine-tuned variants, and widely used off-the-shelf encoder-based detectors. The results show that LoRA fine-tuning consistently improves detection accuracy while substantially reducing false positives, false negatives, and attack success rates. Moreover, lightweight fine-tuned models retain much of the performance of larger counterparts, supporting efficient and scalable deployment. Overall, our findings demonstrate that architectural guardrails combined with targeted model adaptation provide an effective and regulation-aligned approach to securing LLM-integrated systems.

II. Related Works

Prompt injection attacks have emerged as one of the most critical challenges for LLM-integrated systems [18][19]. Prompt injection refers to adversarial techniques in which attackers craft inputs that manipulate model behaviour, causing the LLM to deviate from intended system instructions or safety constraints. Prior work generally categorises prompt injection attacks into two broad classes. The first comprises manually crafted attacks, often relying on prompt engineering strategies such as instruction overriding (e.g., "ignore previous instructions"), context manipulation, escape sequences, and composite attack patterns that combine multiple techniques. The second class includes optimisation-based approaches, such as gradient-driven attacks, which aim to induce specific model outputs by optimising a predefined loss function. These attacks typically seek to maximise attack effectiveness by systematically steering model responses toward adversarial objectives.

Building on the diversity of prompt injection attack characteristics and severity levels, prior work has proposed a wide range of defence mechanisms in both academic and industrial settings [20][21]. A prominent line of research focuses on fine-tuning large language models to improve their robustness against prompt injection. Early approaches, such as [22] fine-tune LLMs including LlaMA [3] and Mistral [23] using structured query formats that explicitly separate system instructions, user inputs, and contextual data into distinct channels. While effective under controlled conditions, subsequent analysis by [24] reveals that such structure-based defences exhibit limited generalisation when confronted with previously unseen attack patterns.

To overcome this limitation, later work reformulates prompt injection defence as a preference optimisation problem. Reference [25] construct dedicated preference datasets and fine-tune instruction-following models such as Mistral-7B-Instruct [23] and LLaMA3-8B-Instruct [3], building on models previously trained via supervised fine-tuning (SFT). These approaches aim to improve robustness by aligning model behaviour with preferred safe responses rather than relying solely on rigid prompt structures.

Complementary fine-tuning-based detection methods treat prompt injection as a classification task. Reference [15] propose DataSentinel, which fine-tunes multiple LLM variants including Mistral-7B, LLaMA2-7B, and LLaMA3-8B-Instruct, to detect prompts contaminated with injected instructions, formulating the problem as a minimax optimisation task. Similarly, [21] introduce DataFilter, which is trained via supervised fine-tuning on synthetically generated injection attacks and removes malicious prompt segments at inference time before forwarding inputs to the backend LLM.

In parallel to generative model-based approaches, several works adopt encoder-based architectures for prompt injection detection. For example, ProtectAI [26] employs a DeBERTa-based classifier trained on aggregated open-source datasets to identify malicious prompts, offering a lightweight alternative to full LLM fine-tuning. Beyond model adaptation, inference-time defences such as [27] explore increasing computational resources during inference, generating multiple candidate responses and selecting outputs aligned with the target task objective.

In parallel with prior works [28] [29][19] [30] that rely on fine-tuning LLMs for prompt injection detection, this paper adopts an architectural defence strategy that operates at the input layer of LLM-integrated systems. By detecting prompt injection attempts before user inputs are forwarded to backend LLMs, the proposed approach provides an additional layer of robustness that is independent of the task-specific model configuration.

III. Methods

This section presents our methodology, covering the training dataset construction, the guardrail-based model architecture, and the experimental setup used to evaluate prompt injection detection.

A. Training Dataset

Due to licensing constraints, the dataset used in this study cannot be publicly released. The training data was aggregated from multiple open-source Hugging Face datasets, including Phare [31], WildGuard [32], and Bitext [33], and subsequently harmonized through a unified preprocessing pipeline to ensure consistency in format and labeling. An auxiliary LLM was employed to enrich the annotations: unsafe prompts were labeled with prompt injection type, supporting evidence, and reasoning, while safe prompts were annotated with evidence only. After cleaning and validation, the final dataset comprises 155,000 samples and is approximately balanced (48.5% safe, 51.5% unsafe). The resulting taxonomy spans a diverse set of injection types, with system instruction override attacks being the most prevalent (21.9% of unsafe samples). Notably, the distribution exhibits a pronounced long-tail pattern, with many injection types occurring infrequently, reflecting the heterogeneity of real-world adversarial prompts and posing challenges for generalization.

B. Model Architecture

The increasing deployment of large language models (LLMs) across diverse application domains has expanded their attack surface, particularly with respect to adversarial inputs designed to manipulate model behaviour. Attackers may attempt to override system instructions or bypass safety constraints, causing LLMs to follow malicious intent rather than the intended user or system objectives. This threat model motivates the design of security-aware architectures capable of detecting and mitigating prompt-level vulnerabilities in real time.

To address this challenge, we propose a governance-oriented guardrail architecture that operates as an intermediary layer between users and backend LLM services. As illustrated in Fig. 1, the guardrail provides prompt-level monitoring and control by evaluating each user prompt before it is forwarded

to the backend task-specific LLM. This design enables early detection of prompt injection attempts at the architectural level, prior to any reasoning or task execution by the core model.

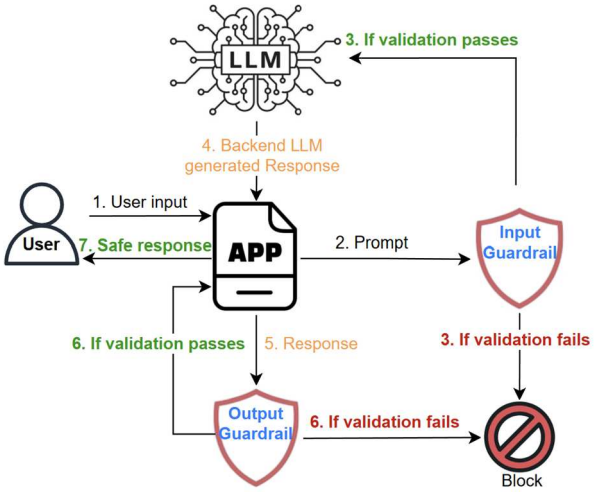

Fig. 1. Architecture of the proposed prompt injection guardrail.

The guardrail first analyses incoming user prompts to assess potential malicious intent, including attempts to override system instructions or circumvent safety policies. Prompts classified as safe are forwarded to the backend LLM, while those identified as unsafe are restricted or blocked according to predefined governance policies. In parallel, the architecture also evaluates LLM-generated outputs to detect residual risks, such as policy circumvention, unsafe content propagation, or model manipulation that may result in misleading or non-compliant responses. This dual-stage assessment strengthens the overall robustness of the system against adversarial prompt manipulation.

The core design principle of the proposed architecture is to leverage the reasoning capabilities of LLMs themselves for prompt injection detection. To this end, we explore multiple configurations, including off-the-shelf LLMs operating in a zero-shot setting, fine-tuned white-box LLM models, and independent services for prompt injection risk detection. This comparative setup enables a systematic evaluation of baseline models against domain-adapted guardrail models.

For the white-box configuration, we fine-tune an LLM using the training dataset. Fine-tuning is performed using a Low-Rank Adaptation (LoRA) strategy, which allows efficient and targeted adaptation of the model's internal representations without modifying the full parameter set. This approach supports scalable deployment and reduces computational overhead while maintaining high detection performance. To further enhance robustness, we employ enriched system prompts that provide explicit instructions for identifying prompt injection patterns and adversarial intent.

To support accountability and auditability, the system attaches a lightweight metadata watermark [34] [35] to each processed interaction, recording the prompt risk assessment outcome at inference time and enabling traceable logging for post-hoc review and compliance verification.

The architecture is designed to support real-time detection of prompt injection attacks. Model performance is evaluated using macro-averaged F1 metrics, with particular emphasis on false positive rate (FPR) and false negative rate (FNR). Minimising false positives is critical to avoid benign prompt misclassification, while reducing false negatives is essential to prevent malicious prompts from being erroneously classified as safe. By combining architectural guardrails with LoRA-based fine-tuning, the proposed system aims to improve prompt injection detection accuracy while maintaining operational efficiency in deployed environments.

C. vLLM-Based Inference with Multiple LoRA Adapter

The proposed architecture is based on a vLLM-backed inference server that dynamically loads multiple LoRA adapters. This design extends a single shared base model with task- or policy-specific LoRA modules at inference time, eliminating the need to deploy separate full model instances for each use case. Consequently, diverse requirements such as prompt injection detection, domain-specific reasoning, or stricter regulatory enforcement can be satisfied within the same serving stack by routing requests to the appropriate adapter. Compared to serving multiple fully fine-tuned models, this approach reduces compute and memory overhead, simplifies operations, and enables rapid iteration and A/B testing at the adapter level. In addition, isolating adaptations within LoRA modules mitigates catastrophic forgetting and supports tenant- or application-specific customization in multi-tenant environments.

D. Experimental Setups

Models. We evaluate LLMs across multiple scales and architectures, including Gemma3-1B/4B/12B/27B [1] and Qwen3-0.6B/4B/14B [2]. For each backbone, we train LoRA fine-tuned variants under identical configurations to assess parameter-efficient adaptation for prompt injection detection. We additionally benchmark encoder-based off-the-shelf detectors, DistilBERT [36], DeBERTa (Deepset [37]), and DeBERTa (ProtectAI [26]) which are commonly used as standalone guard models.

Prompts. We evaluate model performance under two prompting strategies: basic and complex prompts. The basic prompt is a concise instruction (e.g., *"You are a prompt injection detector. Determine whether the given input contains injection attempts or harmful content."*), designed to reflect minimal guidance. In contrast, the complex prompt provides more detailed instructions, including explicit descriptions of injection types, potential attack patterns, and decision criteria. This setting allows us to assess how additional prompt structure and guidance influence detection performance and robustness across models.

Training Details. All fine-tuning is performed on a single NVIDIA GH200 GPU using a lightweight, deployment-oriented setup. Models are trained for one epoch with an effective batch size of 64 (batch size 16, gradient accumulation 4). We use AdamW (8-bit) with a learning rate of 3×10^(-4), cosine scheduling with 5 warm-up steps, and weight decay of 0.01. Mixed-precision training (BF16/FP16) is enabled, and all runs use a fixed random seed (3407).

Metrics. Performance is measured using Precision, Recall, and F1. To capture deployment-relevant trade-offs, we additionally report False Positive Rate (FPR), False Negative Rate (FNR), and Attack Success Rate (ASR), where ASR quantifies the fraction of adversarial prompt injection attempts misclassified as benign.

TABLE I. PERFORMANCE OF PROMPT INJECTION DETECTORS ACROSS LLMS ON THE HELD-OUT DATASET (ALL RESULTS ARE IN PERCENTAGES)

	Prec.	Rec.	F1	FPR	FNR	ASR
Baseline Models Inference with Basic Prompt						
Gemma3-1B	59.06	96.86	73.38	72.35	3.14	36.45
Gemma3-4B	66.39	91.12	76.81	49.74	8.88	28.54
Gemma3-12B	62.37	97.94	76.21	63.71	2.06	31.73
Gemma3-27B	61.65	97.78	75.62	65.58	2.22	32.71
Qwen3-0.6B	56.80	94.80	71.10	77.75	5.15	40.09
Qwen3-4B	69.70	93	79.70	43.54	6.95	24.56
Qwen3-14B	84.8	91.8	87.90	18.31	8.24	13.09
Baseline Models Inference with Complex Prompt						
Gemma3-1B	51.85	99.94	68.28	99.93	0.06	48.15
Gemma3-4B	62.56	93.04	74.82	60.01	6.96	32.49
Gemma3-12B	67.96	78.00	72.64	39.63	22.00	30.48
Gemma3-27B	64.38	88.07	74.38	52.54	11.93	31.47
Qwen3-0.6B	57.80	63.5	60.50	50.02	36.53	43.03
Qwen3-4B	89.00	50.90	64.70	6.782	49.13	28.75
Qwen3-14B	94.40	50.90	66.10	3.24	49.09	27.03
LoRA Fine-tuned Models Inference with Basic Prompt						
Gemma3-1B	98.47	98.13	98.30	1.64	1.87	1.76
Gemma3-4B	99.33	98.29	98.80	0.72	1.71	1.23
Gemma3-12B	97.75	98.57	98.16	2.45	1.43	1.92
Gemma3-27B	99.31	98.85	**99.08**	0.74	1.15	**0.95**
Qwen3-0.6B	97.90	90.70	94.20	2.06	9.27	5.80
Qwen3-4B	99.10	97.80	98.40	0.98	2.25	1.64
Qwen3-14B	98.10	98.70	98.40	2.04	1.29	1.65
LoRA Fine-tuned Models Inference with Complex Prompt						
Gemma3-1B	83.24	97.05	89.61	19.47	2.95	11.23
Gemma3-4B	95.40	94.99	95.19	4.93	5.01	4.98
Gemma3-12B	91.94	93.30	92.61	8.82	6.70	7.72
Gemma3-27B	94.08	95.90	94.98	6.50	4.10	5.26
Qwen3-0.6B	95.60	93.8	94.69	4.19	16.21	10.42
Qwen3-4B	99.70	58.00	73.30	0.21	42.02	21.90
Qwen3-14B	96.60	91.60	94.00	3.50	8.39	6.04
Off-the-shelf Models						
DistilBERT	60.91	88.81	72.26	61.45	11.19	35.38
Deepset	59.86	100	74.89	100	0	40.14
ProtectAI	74.95	3.82	7.28	1.38	96.18	50.56

IV. RESULTS AND DISCUSSION

Table I reveals clear performance gaps between baseline LLMs, LoRA fine-tuned models, and off-the-shelf encoder-based detectors. Overall, LoRA fine-tuning yields substantial and consistent gains across all metrics, markedly improving F1 while sharply reducing false positives (FPR), false negatives (FNR), and attack success rates (ASR).

Baseline LLMs evaluated with basic prompts exhibit high recall but poor precision, leading to excessive false positives. For example, Gemma3-1B attains 96.86% recall but only 59.06% precision, with an FPR of 72.35% and ASR of 36.45%. Similar trends hold across Gemma and Qwen variants, indicating sensitivity to adversarial inputs but weak discrimination of benign prompts. Qwen3-14B performs best among baselines (86.9% accuracy, 87.9% F1), yet still falls short of deployment-grade reliability.

Increasing prompt complexity does not reliably improve baseline robustness. Although recall often increases, this comes at the cost of sharply degraded precision and accuracy. For instance, Gemma3-1B with complex prompts reaches near-perfect recall (99.94%) but suffers an FPR of 99.93% and F1 of 68.28%. In Qwen models, complex prompts substantially raise FNR and ASR, demonstrating that prompt engineering alone is an unstable defense and can even exacerbate vulnerabilities.

In contrast, LoRA fine-tuned models achieve strong and stable performance. With basic prompts, fine-tuned Gemma models reach 98.08–99.05% accuracy with FPR and FNR below 2.5%, while Gemma3-27B attains the best overall performance (99.08% F1, 0.95% ASR). Fine-tuned Qwen models show similar robustness, with Qwen3-4B and Qwen3-14B achieving 98.4% F1 and ASR below 1.7%.

Under complex prompts, fine-tuned models experience moderate degradation. Gemma variants show small F1 reductions (~2 points), whereas Qwen models are more affected; Qwen3-4B drops to 73.3% F1 with an ASR of 21.9%, driven primarily by elevated FNR. This suggests that overly restrictive prompt formulations can suppress true positives even in fine-tuned models, underscoring the need for careful prompt–model co-design.

Comparisons across model scales highlight favorable efficiency, performance trade-offs. Despite large parameter differences, Gemma3-1B achieves 98.3% F1, only marginally below Gemma3-27B, while Qwen3-0.6B retains 94.2% F1 relative to 98.4% for Qwen3-14B. These results indicate that lightweight LoRA-adapted models preserve much of the detection capability of larger models, making them well suited for resource-constrained deployments. Off-the-shelf encoder-based detectors perform substantially worse. DistilBERT achieves only 72.26% F1 with a 35.38% ASR. DeBERTa (Deepset) attains perfect recall but an unusable 100% FPR, while DeBERTa (ProtectAI) shows high precision but extremely low recall (3.82%) and ASR exceeding 50%. These results highlight the limitations of static encoder-based classifiers for prompt injection detection.

In summary, the results demonstrate that LoRA fine-tuning is critical for reliable prompt injection detection, delivering substantial reductions in FPR, FNR, and ASR compared to baseline and off-the-shelf approaches. While increased prompt complexity alone is insufficient—and in some cases harmful—this effect can be attributed to overly restrictive or verbose instructions that bias the model toward conservative decision boundaries, suppressing true positives and increasing false negatives. Such prompts may also introduce ambiguity or dilute salient signals needed for accurate detection. In contrast, targeted model adaptation through LoRA enables robust, efficient, and deployment-ready guardrails for LLM-integrated systems.

V. CONCLUSION

This work investigated prompt injection as a system-level security risk and demonstrated that effective mitigation requires architectural guardrails rather than reliance on backend model alignment alone. Through extensive evaluation across Gemma and Qwen model families, we showed that baseline prompted LLMs and off-the-shelf

encoder-based detectors suffer from unstable precision–recall trade-offs and unacceptably high false positive or attack success rates, limiting their suitability for deployment.

In contrast, LoRA fine-tuned guardrail models consistently achieved strong, deployment-grade performance, substantially reducing false positives, false negatives, and attack success rates across diverse attack categories. Notably, lightweight fine-tuned models retained much of the robustness of larger counterparts, highlighting a practical path toward scalable and cost-efficient deployment. Our results further indicate that increasing prompt complexity alone does not reliably improve robustness and may introduce new vulnerabilities if not carefully designed.

Overall, the findings underscore the importance of prompt-level governance layers combined with targeted model adaptation for securing LLM-integrated systems. By decoupling prompt injection detection from task-specific reasoning models, the proposed approach supports robust, model-agnostic deployment and aligns naturally with emerging regulatory expectations around robustness, cybersecurity, and risk mitigation in LLM-based systems

VI. Future Works and Limitations

This work has several limitations. The training data, aggregated from open-source sources and enriched with LLM-generated annotations, may introduce noise or bias, warranting future expert validation and agreement analysis. Dataset licensing constraints prevent public release, limiting reproducibility and motivating the creation of a shareable or synthetic benchmark. Evaluation on an internal held-out set with standard metrics does not fully capture real-world conditions such as domain shift, adaptive adversaries, multilingual inputs, or the observed sensitivity to prompt complexity, which occasionally increases false negatives and requires further ablation. At the systems level, the vLLM-based deployment with dynamic LoRA adapters raises challenges around routing accuracy and inference latency under load, calling for end-to-end performance analysis and uncertainty-aware routing. While metadata tagging supports auditability, standardized logging and tighter integration with compliance workflows remain open challenges.

References

[1] G. Team and G. Deepmind, “Gemma 3 Technical Report,” pp. 1–25, 2025.

[2] Q. Team, “Qwen3 Technical Report,” pp. 1–35, 2025.

[3] L. Team and A. I. Meta, “The Llama 3 Herd of Models,” pp. 1–92, 2024.

[4] L. Methods, “A Comprehensive Survey on Automatic Text Summarization with Exploration of LLM-Based Methods,” 2025.

[5] S. Dai and C. Xu, “Unifying Bias and Unfairness in Information Retrieval : New Challenges Unifying Bias and Unfairness in Information Retrieval : New Challenges in the LLM Era,” no. March 2025, 2026, doi: 10.1145/3701551.3703478.

[6] O. J. Gstrein, N. Haleem, and A. Zwitter, “General-purpose AI regulation and the European Union AI Act,” vol. 13, 2024.

[7] B. I. Barberá, “OF EXPERTS PROGRAMME AI Privacy Risks & Mitigations Large Language Models (LLMs),” 2025.

[8] K. Greshake, C. Endres, and M. Fritz, Not what you ’ ve signed up for : Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection, vol. 1, no. 1. Association for Computing Machinery, 2023.

[9] F. Perez, “Ignore Previous Prompt : Attack Techniques For Language Models,” no. Figure 1, pp. 1–21, 2022.

[10] E. Debenedetti, J. Zhang, and M. Balunovic, “AgentDojo : A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents,” no. NeurIPS, 2024.

[11] OWASP, “OWASP Top 10 for Large Language Model Applications.” [Online]. Available: https://owasp.org/www-projecttop-%0A10-for-large-language-model-applications

[12] A. Reid, S. O. Callaghan, L. Carroll, and T. Caetano, “Risk Analysis Techniques for,” no. July, 2025.

[13] A. Sekar, M. Agarwal, R. Sharma, A. Tanaka, and J. Zhang, “Zero-Shot Embedding Drift Detection : A Lightweight Defense Against Prompt Injections in LLMs arXiv : 2601 . 12359v1 [cs . CR] 18 Jan 2026,” no. NeurIPS, 2025.

[14] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, “Formalizing and Benchmarking Prompt Injection Attacks and Defenses,” 33rd USENIX Secur. Symp. (USENIX Secur. 24), pp. 1831–1847, 2024.

[15] Y. Liu, Y. Jia, J. Jia, D. Song, and N. Z. Gong, “DataSentinel: A Game-Theoretic Detection of Prompt Injection Attacks,” 2025 IEEE Symp. Secur. Priv., pp. 2190–2208, 2025.

[16] J. Piet et al., “Jatmo: Prompt Injection Defense by Task-Specific Finetuning,” Eur. Symp. Res. Comput. Secur., vol. Cham: Spri, pp. 105–124, 2024.

[17] E. J. Hu et al., “Lora: Low-rank adaptation of large language models.,” ICLR, vol. 1, no. 2, p. 3, 2022.

[18] K. Hung, C. Ko, A. Rawat, I. Chung, W. H. Hsu, and P. Chen, “Attention Tracker : Detecting Prompt Injection Attacks in LLMs,” pp. 2309–2322, 2025.

[19] T. Shi et al., “PromptArmor : Simple yet Effective Prompt Injection Defenses,” arXiv Prepr. arXiv2507.15219, 2025.

[20] H. Li and X. Liu, “InjecGuard: Benchmarking and Mitigating Over-defense in Prompt Injection Guardrail Models,” arXiv Prepr. arXiv2410.22770, 2023.

[21] Y. Wang, S. Chen, R. Alkhudair, B. Alomair, and D. Wagner, “Defending Against Prompt Injection with DataFilter Utility-Security Trade-Off Attack Success Rate,” Proc. 63rd Annu. Meet. Assoc. Comput. Linguist. (Volume 1 Long Pap., no. Llm, pp. 18331–18347, 2025.

[22] S. Chen, U. C. Berkeley, and D. Wagner, “StruQ : Defending Against Prompt Injection with Structured Queries,” 34th USENIX Secur. Symp. (USENIX Secur. 25), no. i, pp. 2383–2400, 2025.

[23] A. Q. Jiang et al., “Mistral 7B,” pp. 1–9.

[24] S. Chen et al., “SecAlign : Defending Against Prompt Injection with Preference Optimization,” Proc. 2025 ACM SIGSAC Conf. Comput. Commun. Secur., pp. 2833–2847, 2025.

[25] Y. Chen, H. Li, Z. Zheng, D. Wu, Y. Song, and B. Hooi, “Defense Against Prompt Injection Attack by Leveraging Attack Techniques,” n Proc. 63rd Annu. Meet. Assoc. Comput. Linguist. (Volume 1 Long Pap., pp. 18331–18347, 2025.

[26] “ProtectAI.” [Online]. Available: https://huggingface.co/protectai/deberta-v3-base-prompt-injection-v2

[27] Y. Liu, Y. Wang, Y. Jia, J. Jia, and N. Z. Gong, “SecInfer : Preventing Prompt Injection via Inference-time Scaling,” arXiv Prepr. arXiv2509.24967, pp. 1–32, 2025.

[28] J. Piet et al., “Jatmo: Prompt Injection Defense by Task-Specific Finetuning BT - Computer Security – ESORICS 2024,” J. Garcia-Alfaro, R. Kozik, M. Choraś, and S. Katsikas, Eds., Cham: Springer Nature Switzerland, 2024, pp. 105–124.

[29] Y. Jia, Z. Shao, Y. Liu, J. Jia, D. Song, and N. Z. Gong, “A Critical Evaluation of Defenses against Prompt Injection Attacks,” arXiv Prepr. arXiv2505.18333, pp. 1–15, 2025.

[30] W. Hackett, L. Birch, S. Trawicki, N. Suri, and P. Garraghan, “Bypassing LLM Guardrails : An Empirical Analysis of Evasion Attacks against Prompt Injection and Jailbreak Detection Systems,” arXiv Prepr. arXiv2504.11168, pp. 101–114, 2025.

[31] https://huggingface.co/datasets/giskardai/phare

[32] https://huggingface.co/allenai/wildguard

[33] https://huggingface.co/datasets/bitext/Bitext-retail-banking-llm-chatbot-training-datase

[34] B. Rijsbosch, G. Van Dijck, and K. Kollnig, Missing the Mark : Adoption of Watermarking for Generative AI Systems in Practice and Implications under the new EU AI Act, vol. 1, no. 1. arXiv, 2025.

[35] M. S. Labs, “How Good is Post-Hoc Watermarking With Language Model Rephrasing?,” arXiv Prepr. arXiv2512.16904, pp. 1–24, 2025.

[36] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT , a distilled version of BERT : smaller , faster , cheaper and lighter,” arXiv Prepr. arXiv1910.01108, pp. 2–6, 2019.

[37] https://huggingface.co/datasets/deepset/prompt-injections, “Deepset.”