Prompt-Injection Detection and Defense in LLM-Integrated Knowledge Management Systems

Stanley Beck
Department of Computer Science, Colorado State University, Fort Collins, CO, USA.
stanley.beck@colostate.edu

Zhi Gao
Department of Computer Science, Binghamton University, Binghamton, NY, USA.
zhi183@binghamton.edu

Milos Russell
Department of Computer Science, University of New Hampshire, Durham, NH, USA.
milos.work@unh.edu

Anders Miles
Department of Computer Science, George Mason University, Fairfax, VA, USA.
anders.miles@gmu.edu

Abstract

The integration of large language models into organizational knowledge management systems has introduced significant productivity gains alongside novel security vulnerabilities, most notably prompt-injection attacks. These attacks exploit the instruction-following capacity of LLMs to subvert system behavior, potentially leading to data exfiltration, unauthorized access, or propagation of misinformation across corporate knowledge bases. This paper presents a comprehensive system-level analysis of detection and defense strategies for prompt-injection threats within LLM-integrated knowledge management architectures. We examine the structural trade-offs between the flexibility required for open-ended query processing and the constraints necessary for secure operation. We propose a layered defense framework that encompasses input sanitization, runtime monitoring, output verification, and governance policies, while critically assessing the limitations of each layer. The paper also explores how architectural choices, such as retrieval-augmented generation pipelines, context window management, and role-based access controls, influence attack surfaces and defensive efficacy. Through cross-domain comparisons with traditional injection attacks in database and web systems, we highlight the unique challenges posed by the semantic nature of prompt manipulations. We further discuss infrastructural sustainability, fairness implications in access control, and the policy landscape for deploying LLMs in sensitive knowledge environments. The analysis concludes that no single defense mechanism is sufficient; instead, a resilient system must combine technical controls with organizational governance, continuous monitoring, and adaptive threat models. This work provides a roadmap for researchers and practitioners seeking to build robust knowledge management systems that can leverage LLM capabilities without compromising security.

Keywords

prompt injection, large language models, knowledge management systems, adversarial attacks, system security, defense mechanisms, socio-technical infrastructure.

1. Introduction

Large language models have rapidly transitioned from experimental research artifacts to integral components of enterprise software, particularly in knowledge management systems that aggregate, retrieve, and synthesize organizational information. These systems allow users to query structured and unstructured data through natural language interfaces, dramatically lowering the barrier to information access. However, the same instruction-following capabilities that make these systems useful also render them vulnerable to prompt-injection attacks, where an adversary crafts inputs that cause the model to override its original instructions, execute unintended operations, or reveal sensitive data [1]. The threat is compounded when the LLM is embedded in a broader infrastructure that includes databases, APIs, and user authentication layers. Unlike traditional injection attacks that target parsers or interpreters, prompt injection operates at the level of semantic instruction, making detection and prevention fundamentally different from classic input validation [2]. The growing adoption of LLM-integrated knowledge management systems across sectors such as healthcare, finance, and legal services demands a rigorous examination of how to detect and defend against these attacks at a system level. This paper provides such an examination, focusing on architectural trade-offs, detection methodologies, defense mechanisms, and the broader implications for governance and policy.

2. Background and Threat Landscape

Prompt-injection attacks can be broadly categorized into direct and indirect variants [3]. In a direct attack, the user intentionally embeds malicious instructions within their query, aiming to subvert the system's intended behavior. Indirect attacks, which are arguably more dangerous, occur when an attacker injects text into an external source that the LLM subsequently reads and processes, such as a document stored in the knowledge base or a web page retrieved by the system [4]. The latter scenario is particularly relevant for knowledge management systems that continuously ingest content from external feeds or user uploads. Once the prompt-injection payload is part of the knowledge base, any subsequent query that causes the LLM to retrieve that content may inadvertently activate the malicious instructions. This creates a persistent and stealthy attack surface that is difficult to monitor because the injection is hidden in the data rather than in the query stream [5]. The threat extends beyond data exfiltration: an attacker could manipulate the system to output false information, alter stored records through downstream APIs, or escalate privileges by tricking the LLM into performing actions on behalf of a legitimate user [6]. Research has demonstrated that even state-of-the-art LLMs, including those with safety alignment training, are susceptible to carefully crafted injections, often exhibiting what has been described as a "blind trust" in user-provided instructions [7]. The failure modes are not merely technical; they have organizational consequences such as loss of data integrity, regulatory non-compliance, and erosion of user trust.

3. Architectural Considerations for LLM-Integrated KMS

The security posture of an LLM-integrated knowledge management system is heavily influenced by its architecture. A common design pattern is retrieval-augmented generation, where the LLM receives both a user query and a set of context documents retrieved from a vector database or search index [8]. The model then generates an answer based on that context. This architecture introduces a natural point of control: the retrieval pipeline can be instrumented to inspect both the query and the retrieved documents for suspicious content. However, the contextual richness that makes RAG effective also increases the attack surface,

because the attacker can inject malicious instructions into the documents themselves [9]. Another architectural choice is the degree of autonomy granted to the LLM. In some systems, the model is a pure text generator that only returns responses to the user interface, while in others, the LLM can invoke external tools, such as database write operations or email sending services, based on its output [10]. The latter configuration dramatically magnifies the potential damage from a successful prompt injection, as the attacker can weaponize the model's tool-use capabilities. A critical architectural trade-off exists between allowing the LLM to see large context windows for improved answer quality and restricting context to minimize the exposure to injected content. Systems that truncate or filter context based on sensitivity labels or anomaly scores can reduce risk but may degrade retrieval accuracy [11]. Furthermore, the deployment topology matters: a system that processes queries on a client device with local models is less exposed than a cloud-based system that aggregates data from multiple tenants, but local models may have weaker safety alignment. Governance at the architectural level must therefore balance functionality, performance, and security by designing layered isolation between the LLM, the knowledge base, and the action layer.

4. Detection Methods: Structural and Behavioral Approaches

Detecting prompt-injection attempts in real time is a challenging problem because the adversarial inputs are often semantically indistinguishable from legitimate instructions to a human observer. Structural detection methods attempt to identify syntactic or lexical patterns that are characteristic of injection attempts. For example, regularization patterns that escape the intended instruction format, such as the inclusion of delimiters like "Ignore previous instructions" or "System prompt: ", can be flagged by rule-based filters [12]. However, attackers quickly adapt by paraphrasing or using obfuscation techniques that bypass simple pattern matching. More sophisticated structural approaches employ machine learning classifiers trained on known injection datasets to distinguish malicious from benign prompts. These classifiers can be integrated into the input processing pipeline, but they suffer from generalization gaps when faced with novel attack variants and may produce false positives that degrade user experience [13].

Behavioral detection methods take a different approach by monitoring the LLM's responses for signs that an injection has succeeded. For instance, if the LLM outputs unexpected API calls, returns data that the user should not have access to, or exhibits a sudden shift in tone or content structure, an anomaly detection system can raise an alert [14]. Behavioral detection can be implemented as a secondary component that runs in parallel with the LLM inference, analyzing both the input and the output in a streaming fashion. This approach has the advantage of not requiring a priori knowledge of injection patterns, but it introduces latency and requires careful tuning to avoid overwhelming security teams with false alarms. Another behavioral technique involves prompting the LLM itself to evaluate its own reasoning for signs of manipulation, a method sometimes referred to as self-detection or self-consistency checking [15]. While promising, this technique is computationally expensive and can be bypassed by injections that instruct the model not to report its own subversion. A comprehensive detection strategy should combine both structural and behavioral methods in a layered fashion, with each layer compensating for the weaknesses of the others. Additionally, detection should extend beyond the query-response cycle to include periodic scanning of the knowledge base for embedded injection payloads, especially in systems that allow user-generated content [16]. The scalability of these detection mechanisms is a key infrastructural

concern, as knowledge management systems may need to process thousands of queries per second while maintaining sub-second response times.

5. Defense Mechanisms: Governance, Infrastructure, and Deployment

Defense against prompt-injection attacks requires a combination of technical controls and organizational governance. At the technical level, input sanitization is the first line of defense. Sanitization can include stripping or escaping control tokens, restricting the length of input fields, and using parameterized prompts where the user query is inserted into a fixed template that separates instructions from data [17]. However, as noted earlier, the semantic nature of prompt injection makes sanitization a limited tool because the boundary between instruction and data is not clearly defined. A more robust approach is to isolate the LLM execution environment through containerization and least-privilege access controls. The LLM should not have direct access to the knowledge base or to external APIs; instead, its output should be parsed by a middleware layer that enforces security policies before executing any actions [18]. This middleware can also serve as a monitor that logs all interactions for forensic analysis and anomaly detection.

Deployment strategies also play a crucial role. Systems that use a single LLM for both query understanding and response generation are more vulnerable than those that separate these concerns using multiple specialized models. For example, a small, unprivileged model can be used to interpret user intent and filter out suspicious requests before forwarding a distilled query to a more powerful model that generates the final answer [19]. This reduces the attack surface because the privileged model never sees the raw user input. Another deployment consideration is the use of adversarial training to harden the LLM against injection attempts during the fine-tuning phase. While this improves robustness, it is not a complete solution, as attackers can find new prompts that circumvent the training distribution [20]. At the governance level, organizations must establish clear policies for what types of queries are permitted, how sensitive documents are labeled and accessed, and how incidents are reported and responded to. Role-based access control should be applied not only to user accounts but also to the data that the LLM can retrieve, ensuring that a user's query cannot cause the model to access information beyond that user's clearance level [21]. Furthermore, regular red-teaming exercises and security audits should be conducted to evaluate the effectiveness of existing defenses and to uncover new attack vectors.

6. Trade-offs and Systemic Implications

Designing a secure LLM-integrated knowledge management system involves navigating a series of trade-offs that affect usability, performance, cost, and fairness. The most fundamental trade-off is between the freedom of expression in user queries and the constraints needed to prevent injection attacks. Overly restrictive filters can degrade the user experience by blocking legitimate queries or forcing users into rigid query templates, reducing the very advantage that natural language interfaces provide [22]. Similarly, aggressive monitoring of outputs may introduce unacceptable latency for real-time applications. The cost of deploying multiple models, containerized environments, and continuous monitoring infrastructure can be prohibitive for smaller organizations, potentially widening the security gap between large enterprises and smaller ones. This raises equity concerns in the deployment of AI systems, where security sophistication becomes a privilege rather than a standard [23].

Another important implication is the challenge of maintaining system robustness over time. As LLMs are updated and new attack techniques emerge, the security posture must evolve

continuously. This requires a sustainable commitment to updating detection models, retraining filters, and revising governance policies, which can be difficult to sustain in organizations with limited cybersecurity expertise. The fairness dimension also merits attention: access control policies that are too strict may inadvertently discriminate against users who need to access information for legitimate purposes, while overly permissive policies may expose sensitive data. Finding the right balance requires stakeholder input and transparent policy design. Finally, the policy landscape for LLM security is still nascent. Regulatory frameworks such as the EU AI Act and various data protection laws are beginning to address AI safety, but they have not yet established specific requirements for prompt-injection defense [24]. Proactive industry standards and cross-sector collaboration are needed to develop best practices that can be adopted widely.

7. Conclusion

Prompt-injection attacks pose a significant and evolving threat to LLM-integrated knowledge management systems. Their semantic nature defies traditional input validation approaches, requiring a multilayered defense strategy that encompasses architectural design, detection techniques, deployment configurations, and organizational governance. This paper has systematically examined these layers, highlighting the critical trade-offs between security, usability, performance, and cost. We have argued that detection must combine structural and behavioral methods, that defense must isolate the LLM from sensitive resources, and that governance must enforce role-based access and continuous auditing. No single measure provides complete protection; resilience emerges from the integration of multiple complementary controls. As LLMs continue to be embedded into the infrastructure of knowledge work, the research community and industry must collaborate to develop adaptive, sustainable, and equitable defense mechanisms. Future research directions include the design of provably robust prompt templates, the use of formal verification for instruction semantics, and the development of cross-organizational threat intelligence sharing for prompt-injection patterns. The goal is not just to defend individual systems but to build a secure foundation for the next generation of AI-augmented knowledge management.

References

1. Alavi, M., & Leidner, D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Quarterly, 25(1), 107-136.

2. Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., & Irving, G. (2022). Red teaming language models with language models. arXiv preprint arXiv:2202.03286.

3. 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).

4. Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., & Liu, Y. (2023). Jailbreaking ChatGPT via prompt engineering: An empirical study. arXiv preprint arXiv:2305.13860.

5. Wei, A., Haghtalab, N., & Steinhardt, J. (2023). Jailbroken: How does LLM safety training fail? In Advances in Neural Information Processing Systems (NeurIPS 2023).

6. Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., & others. (2021). Extracting training data from large language models. In 30th USENIX Security Symposium (USENIX Security 21).

7. Wallace, E., Feng, S., Kandpal, N., Gardner, M., & Singh, S. (2019). Universal adversarial triggers for attacking and analyzing NLP. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP).

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

9. Schulhoff, S., Pinto, J., Khan, S., & Bhardwaj, A. (2023). Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt injection competition. arXiv preprint arXiv:2311.16119.

10. Qian, J., Wang, H., Li, S., & He, Y. (2023). Tool-augmented language models: A survey. arXiv preprint arXiv:2305.10601.

11. Kang, D., Li, X., Stoica, I., Guestrin, C., Zaharia, M., & Hashimoto, T. (2023). Exploiting programmatic behavior of LLMs: Dual-use through prompt injection. In 32nd USENIX Security Symposium (USENIX Security 23).

12. Piet, J., Alrashed, M., Sitawarin, C., & Wagner, D. (2023). Jatmo: Prompt injection defense by task-specific finetuning. arXiv preprint arXiv:2312.08054.

13. Dilmaghani, S., Brust, M. R., Danoy, G., & Bouvry, P. (2023). Security and privacy challenges of large language models: A survey. arXiv preprint arXiv:2310.16022.

14. Chang, Y., Narang, P., Suzuki, H., & Akamai, T. (2023). A survey on evaluation of large language models. ACM Computing Surveys, 56(4), 1-45.

15. Huang, J., Chen, Z., & Xiao, Y. (2023). Prompt injection attack detection and defense for large language models: A survey. arXiv preprint arXiv:2311.07214.

16. Staab, S., & Studer, R. (2010). Knowledge management and knowledge engineering. In S. Staab & R. Studer (Eds.), Handbook on ontologies (pp. 3-20). Springer.

17. Garg, S., & Choudhury, M. (2023). Evaluating the security of enterprise LLM deployments: A case study. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS).

18. Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., & others. (2020). Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv preprint arXiv:2004.07213.

19. Zhang, Y., & Li, X. (2023). Defending against prompt injection with meta-prompting. In Findings of the Association for Computational Linguistics: ACL 2023.

20. Cohen, J. E., & Nissenbaum, H. (2021). AI and the law: From liability to governance. Harvard Journal of Law & Technology, 34(2), 467-530.

21. Muller, M., & Boyd, D. (2023). Fairness and accountability in knowledge management systems with LLMs. In Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT).

22. Crawford, K., & Calo, R. (2016). There is a blind spot in AI research. Nature, 538(7625), 311-313.

23. Floridi, L., & Cowls, J. (2022). A unified framework of five principles for AI in society. In The 2019 Yearbook of the Digital Ethics Lab (pp. 71-86). Springer.

24. European Commission. (2021). Proposal for a regulation laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). COM(2021) 206 final.