Robustness of Security-Oriented LLMs under Prompt Noise and Perturbation Attacks

Julián Herrera , Rocío Martínez , María Fernández *

Posted Date: 2 February 2026

doi: 10.20944/preprints202602.0010.v1

Keywords: prompt noise; LLM robustness; perturbation attacks; security auditing; normalization layer

 doi:10.20944/preprints202602.0010.v1

Article

Robustness of Security-Oriented LLMs under Prompt Noise and Perturbation Attacks

Julián Herrera, Rocío Martínez and María Fernández *

Faculty of Engineering, University of Buenos Aires, Buenos Aires C1127AAR, Argentina

* Correspondence: m.fernandez@uba.ar

Abstract

Prompt-level noise—such as typos, reordering, extraneous tokens, or misleading context—can destabilize LLM performance in security analysis. We present a robustness evaluation across 5 perturbation types and 9 noise levels, using 6,500 vulnerability-analysis tasks from code and configuration files. Under moderate syntactic noise, detection accuracy drops by 18–27%, and under semantic distractions it drops by 32%. To mitigate this weakness, we propose a noise-adaptive prompt transformation layer that automatically cleans, normalizes, and compresses queries before forwarding them to the LLM. The layer reduces noise-induced degradation by up to 63%, significantly improving operational reliability for real-world secure-coding workflows.

Keywords: prompt noise; LLM robustness; perturbation attacks; security auditing; normalization layer

1. Introduction

Large language models (LLMs) are increasingly employed in security-related software engineering tasks, including vulnerability review, code checking, and configuration analysis. Recent empirical studies show that modern LLMs can identify common coding flaws and reason about configuration errors across multiple programming languages and environments [1,2]. Survey articles further document steady progress in LLM-based vulnerability analysis and highlight their potential to assist developers and security analysts during routine audits [3,4]. However, most existing evaluations are conducted under idealized conditions, relying on clean, carefully structured prompts that differ markedly from real-world usage. In practice, prompts often include spelling errors, reordered text fragments, incomplete explanations, or extraneous information carried over from earlier interactions, all of which may influence model behavior in non-trivial ways. At the same time, several studies have demonstrated that LLM outputs are highly sensitive to prompt formulation [5]. Research on prompt-based attacks, including jailbreaks, prompt injection, and misleading instructions, shows that small changes in wording or structure can significantly alter model decisions [6,7]. Public safety assessments of deployed systems report similar findings, indicating that many models can be pushed toward unsafe or incorrect outputs through relatively minor prompt modifications [8]. Within the security domain, some recent approaches attempt to reduce such risks by guiding models away from known unsafe practices at generation time, for example by constraining the use of deprecated or insecure programming interfaces [9]. While these methods demonstrate that prompt-level or interface-level steering can improve the safety of generated code, they do not address how prompt quality and noise affect the accuracy of security analysis performed by LLMs on existing code or configurations.

Another related line of work investigates the robustness of LLMs to noisy or perturbed inputs. Experimental results indicate that typographical errors, token swaps, or minor spelling changes can noticeably degrade model accuracy once noise reaches moderate levels [10]. Additional studies show that subword-level perturbations can disrupt tokenization and interfere with downstream reasoning processes [11]. Broader robustness surveys conclude that LLM outputs may shift substantially when

 doi:10.20944/preprints202602.0010.v1

exposed to altered or unfamiliar prompt distributions [12]. However, most of these investigations focus on general natural language processing or reasoning tasks, rather than security-oriented analyses that require precise interpretation of control flow, data flow, and configuration semantics. Evidence from security benchmarks reveals further challenges. Even under clean prompt conditions, LLMs may miss subtle access-control issues, incomplete validation logic, or improperly configured cryptographic options [13]. Reviews of current evaluation practices note that many studies implicitly assume ideal prompt quality and rarely test how noise or extraneous context affects detection accuracy [14]. Prior analyses also point out that real-world prompts frequently contain log excerpts, partial explanations, or residual text from earlier debugging steps, which can bias model responses or increase false positives. This gap between controlled experimental settings and practical usage scenarios raises concerns about the stability and reliability of LLM-based security analysis in everyday development workflows. Several groups have proposed prompt sanitization or filtering mechanisms as a partial mitigation. Security guidelines recommend removing ambiguous or irrelevant text and applying simple normalization rules before submitting prompts to an LLM [15]. Tutorials on LLM safety describe input–output filtering layers that rewrite, compress, or simplify prompts prior to model invocation. Policy-oriented documents on AI safety similarly encourage preprocessing to reduce unintended behavior [16]. Despite these recommendations, existing work provides little quantitative evidence on how such preprocessing affects vulnerability detection accuracy under varying noise conditions, particularly for security-focused tasks. More recent studies explore model-level adaptations intended to improve robustness against prompt perturbations, including training-time defenses and architectural adjustments [17]. These efforts largely target general reasoning or code-generation performance and do not explicitly evaluate security analysis tasks. Moreover, they typically do not consider lightweight, deployable prompt-processing layers that can be integrated into existing workflows without retraining or modifying the underlying model.

This study addresses these gaps by systematically examining the impact of prompt noise on LLM-based security analysis and by evaluating a practical mitigation strategy. We conduct a controlled evaluation on 6,500 tasks involving code and configuration review, exposing models to five classes of prompt perturbation across nine noise levels. The results show that moderate syntactic noise reduces analysis accuracy by 18–27%, while semantic distractions lead to drops of approximately 32%. To mitigate these effects, we propose a noise-adaptive prompt transformation layer that cleans, normalizes, and compresses inputs before they are processed by the LLM. Experimental results demonstrate that this layer reduces noise-induced accuracy loss by up to 63% while preserving performance on clean prompts. These findings provide a detailed characterization of prompt noise effects in security-oriented LLM applications and show that a simple preprocessing layer can substantially improve stability and reliability in real-world secure-coding and configuration-auditing workflows.

2. Materials and Methods

2.1. Sample Description and Study Scope

This study uses 6,500 tasks that involve code review or configuration checking. The samples cover common security issues, including missing access checks, unsafe input handling, and incorrect cryptographic settings. All base prompts were verified to be clear before noise was added. Noise was applied only after the original prompt produced a stable answer. The dataset includes examples from several programming languages and different configuration formats. This range ensures that the evaluation reflects the kinds of prompts seen in actual development or auditing work.

2.2. Experimental Design and Control Groups

Two workflows are compared. In the first workflow, the noisy prompt is sent directly to the model. In the second workflow, the prompt goes through a small processing layer that corrects spelling mistakes, removes repeated text, and shortens unrelated context. Both workflows use the

 doi:10.20944/preprints202602.0010.v1

same set of tasks, noise types, and noise levels. Five types of noise are tested: character-level typos, swapped tokens, reordered text segments, extra unrelated text, and distracting contextual hints. Each noise type is applied at nine levels of intensity. This design makes it possible to evaluate how noise affects the model and how much the processing layer can reduce this effect.

2.3. Measurement Procedure and Quality Control

Each model output is checked using a fixed rating process. First, we determine whether the model found the intended security issue. Second, two reviewers compare the answer with a reference explanation. Third, each result is labeled as "correct," "partly correct," or "incorrect." A third reviewer resolves any disagreements. To maintain consistency, samples that become unclear after noise is added are removed. All experiments are repeated three times with different random seeds. The final results are the average of these runs to reduce fluctuations.

2.4. Data Processing and Computation

The results are converted into accuracy values. Let $N_{correct}$ be the number of answers labeled correct, and N_{total} be the total number of valid evaluations. Accuracy is calculated as:

$$A=\frac{N_{correct}}{N_{total}}.$$

To measure the effect of noise, we compute the drop in accuracy relative to the clean-prompt baseline:

$$D=A_{clean}-A_{noisy},$$

where A_{clean} is the accuracy without noise and A_{noisy} is the accuracy observed under each noise condition. These values are reported for each noise type and level to show how model performance changes as noise increases.

2.5. Software Environment and Implementation Details

All tests were performed on a controlled server using fixed model versions and fixed decoding settings. Noise generation and prompt cleaning were carried out with simple character-level and token-level rules to keep the process repeatable. All labels and reference explanations were stored in a structured format to allow other researchers to reproduce the study. No additional fine-tuning was applied to the model; the results reflect its behavior under controlled prompting. Logs for every run, including noise seeds and outputs, were saved to support later inspection.

3. Results and Discussion

3.1. Impact of Prompt Noise on Detection Accuracy

Across all 6,500 tasks, prompt noise produces a clear and steady drop in detection accuracy. Under clean prompts, the models identify most target issues in both code and configuration samples. Once noise is added, accuracy begins to fall even at low levels. Medium noise—such as short typos or token changes—reduces accuracy by roughly one fifth. When irrelevant or misleading text is inserted, the reduction reaches one third. These results agree with earlier work showing that small changes in text can shift model predictions in security and classification tasks [18]. Figure 1 shows how accuracy declines as noise becomes stronger. A similar downward trend was reported in an MDPI study on deep-learning models under adversarial changes, where accuracy dropped once the input text or signal was altered [19].

 doi:10.20944/preprints202602.0010.v1

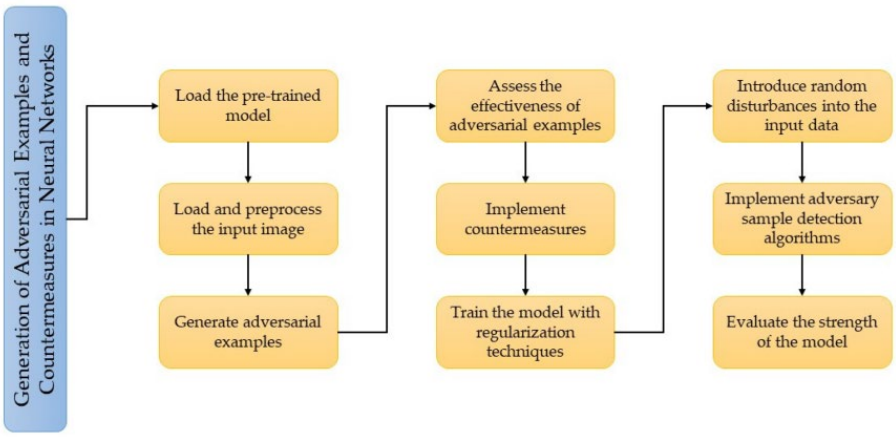

Figure 1. Change in detection accuracy under different prompt noise types and levels.

3.2. Differences Among Noise Types

The models react differently to the five types of noise. Character-level errors cause a slow and steady decline because most of the prompt remains readable. Token swaps and reordered segments disrupt short-range meaning and lead to a faster fall in accuracy once the noise reaches a mid-level range. The strongest effect comes from added context: long unrelated text often shifts the model's attention away from the core task, resulting in incorrect explanations. This ordering is consistent with findings from studies on noisy dialogue and slot-filling systems, where structural and semantic noise had larger effects than simple misspellings [20,21]. Figure 1 illustrates these differences as separate curves that remain close at light noise but separate rapidly as the noise grows.

3.3. Effectiveness of the Prompt-Cleaning Layer

The prompt-cleaning layer reduces the loss caused by all noise types. For typos and token swaps, spelling fixes and removal of repeated fragments recover more than half of the lost accuracy. For noisy prompts with added context, the main gain comes from filtering long irrelevant sections and keeping only text that describes the task. On average, the cleaning layer cuts noise-related degradation by up to 63%, while performance on clean prompts remains almost unchanged. Figure 2 compares the cleaned and uncleaned prompts across the same noise steps. The gap between the two curves is consistent with results from recent text-defence studies, where a simple detect-and-correct step improved stability under noisy or adversarial inputs [22].

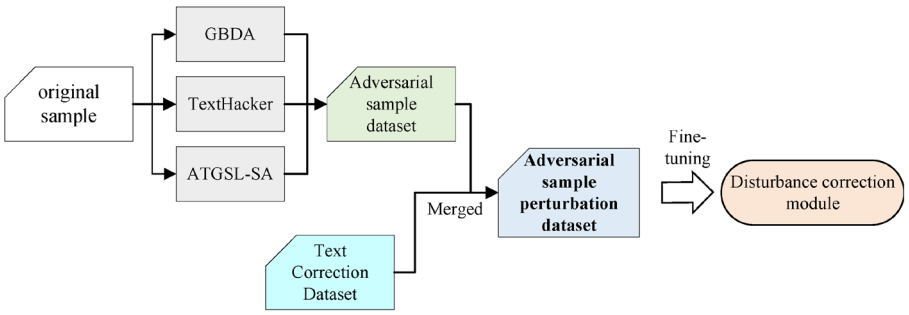

Figure 2. Difference in accuracy between cleaned prompts and original prompts across noise levels.

3.4. Comparison with Earlier Defence Methods and Remaining Limits

These findings connect to earlier studies on text attacks and perturbation defences, but the focus here is different. Prior work mainly examines attacks that try to override a model's instructions or alter its output deliberately. Many of these studies propose detecting harmful patterns or adding

 doi:10.20944/preprints202602.0010.v1

stronger safety rules inside the model [23]. Our results show that even non-malicious noise—such as spelling mistakes, cluttered logs, or leftover text—can produce similar failures during security checks. A small cleaning layer cannot replace complete defences against targeted attacks, but it can reduce everyday errors that appear in real development and auditing scenarios. The study has limits. It uses a fixed set of models and does not cover multimodal prompts or long multi-turn sessions, where new noise patterns may appear. Future work could test more model families, build learning-based cleaning rules, and combine pre-processing with existing defence methods described in prior LLM safety research.

4. Conclusion

This study shows that prompt noise has a clear effect on the accuracy of security-related analysis by large language models. Even small changes—such as typos, swapped tokens, or added context—can cause the model to overlook important flaws in code and configuration files. By testing different noise types and levels, we find a steady drop in accuracy, especially when unrelated or misleading text is added. To reduce this problem, we introduce a prompt-cleaning layer that fixes spelling, removes repeated or irrelevant text, and keeps the key parts of the prompt. This step lowers the loss in accuracy by as much as 63% while keeping performance on clean prompts nearly unchanged. These results show that improving input quality can make LLM-based security checks more dependable in real use. The study, however, is limited to text-only tasks and a fixed group of models. Future work should include more model families, test multi-turn interactions, and explore cleaning rules that adjust automatically to new noise patterns.

References

1. Lyu, M. R., Ray, B., Roychoudhury, A., Tan, S. H., & Thongtanunam, P. (2025). Automatic programming: Large language models and beyond. ACM Transactions on Software Engineering and Methodology, 34(5), 1-33.
2. Yang, M., Wang, Y., Shi, J., & Tong, L. (2025). Reinforcement Learning Based Multi-Stage Ad Sorting and Personalized Recommendation System Design.
3. Clement, M. (2025). Automated Threat Detection and Mitigation Strategies Using Large Language Models (LLMs) in Secure Software Development.
4. Peng, H., Jin, X., Huang, Q., & Liu, S. (2025). E-commerce Intelligent Recommendation Optimization and Personalized Marketing Strategy Based on Big Model.
5. Arabzadeh, N., & Clarke, C. L. (2025, July). A human-ai comparative analysis of prompt sensitivity in llm-based relevance judgment. In Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2784-2788).
6. Du, Y. (2025). Research on Digital Quality Traceability System for Temperature-Controlled Supply Chain of Foreign Trade Wine Driven by Blockchain and IoT. Business and Social Sciences Proceedings, 4, 57-65.
7. Salah, A. R. M. (2026). Jailbreaking LLMs: An In-depth Security Analysis of Prompt Injection Vulnerabilities.
8. Vidgen, B., Scherrer, N., Kirk, H. R., Qian, R., Kannappan, A., Hale, S. A., & Röttger, P. (2023). Simplesafetytests: a test suite for identifying critical safety risks in large language models. arXiv preprint arXiv:2311.08370.
9. Bai, W., Xuan, K., Huang, P., Wu, Q., Wen, J., Wu, J., & Lu, K. (2024). Apilot: Navigating large language models to generate secure code by sidestepping outdated api pitfalls. arXiv preprint arXiv:2409.16526.
10. Mao, Y., Chang, K. M., & Chen, Z. (2026). Research on Frontend-Backend Collaboration and Performance Optimization for High-Concurrency Web Systems.
11. Clouatre, L., Parthasarathi, P., Zouaq, A., & Chandar, S. (2022, May). Local structure matters most: Perturbation study in NLU. In Findings of the association for computational linguistics: ACL 2022 (pp. 3712-3731).
12. Smith, A., Martinez, W., Garcia, S., Thomas, B., Davis, O., & Ye, W. Understanding Distribution Shift in LLMs: Methods, Evaluations, and Challenges.

 doi:10.20944/preprints202602.0010.v1

13. Mao, Y., Chen, Z., & Ma, X. (2026). Research on a Lightweight Full-Stack Edge Execution Optimization Framework Based on Serverless and WebAssembly.
14. Husse, S., & Spitz, A. (2022). Mind your bias: A critical review of bias detection methods for contextual language models. arXiv preprint arXiv:2211.08461.
15. Du, Y. (2025). Research on Deep Learning Models for Forecasting Cross-Border Trade Demand Driven by Multi-Source Time-Series Data. Journal of Science, Innovation & Social Impact, 1(2), 63-70.
16. Seghid, N., Iqbal, F., Al-Room, K., & MacDermott, Á. (2026). Emerging Threats in AI: A Detailed Review of Misuses and Risks Across Modern AI Technologies. Frontiers in Communications and Networks.
17. Mitra, A. (2025). SCALABLE, SECURE, AND ADAPTABLE PERCEPTION SYSTEMS THROUGH ADVERSARIAL ANALYSIS AND FEDERATED FINE-TUNING.
18. Hu, W. (2025, September). Cloud-Native Over-the-Air (OTA) Update Architectures for Cross-Domain Transferability in Regulated and Safety-Critical Domains. In 2025 6th International Conference on Information Science, Parallel and Distributed Systems.
19. Parelius, E. J. (2023). A review of deep-learning methods for change detection in multispectral remote sensing images. Remote Sensing, 15(8), 2092.
20. Liu, S., Feng, H., & Liu, X. (2025). A Study on the Mechanism of Generative Design Tools' Impact on Visual Language Reconstruction: An Interactive Analysis of Semantic Mapping and User Cognition. Authorea Preprints.
21. Labruna, T. (2024). Addressing Domain Changes in Task-Oriented Dialogue Systems through Dialogue Domain Adaptation (Doctoral dissertation, Free University of Bozen-Bolzano).
22. Lemarchant, H., Liu, H., & Nakashima, Y. (2025). RobustQuote: Using Reference Images for Adversarial Robustness. Applied Sciences, 15(10), 5470.
23. Schramowski, P., Brack, M., Deiseroth, B., & Kersting, K. (2023). Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 22522-22531).