



Enterprise AI Prompt Screening System

Abhay Gaidhani¹, Prathmesh Ahire¹, Devashree Jadhav¹, Pooja Gamane¹, Virendra Rathod¹*Under the Guidance of Prof. Abhay Gaidhani*

¹Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India
abhay.gaidhani@sitrc.org, prathmeshahire55@gmail.com, devashreebjadhav@gmail.com, poojagamane@gmail.com,
rathodvirendra18@gmail.com

Corresponding author-Abhay Gaidhani

DOI: 10.5281/zenodo.23298363

Abstract: Generative Artificial Intelligence (AI) tools are currently being widely used in business environments, but the prompts employed are unstructured text that can include personal data, financial information, credentials, or confidential information about an organization. There is also a risk involved in prompt injection, since it enables user instructions to circumvent the existing rules or reveal the system's configuration. In most situations, there is no technical method of controlling access between the user and the model. The present paper introduces an Enterprise AI Prompt Screening System which features a deterministic and explainable screening phase before a large language model that is run locally. The system makes use of Spring Boot, Spring Security, Spring AI, React, and MySQL; it identifies sensitive entities by means of Java recognisers based on rules, detects signals indicating that direct prompt injection is occurring, calculates a weighted Prompt Risk Score, and then carries out one of five policy actions: Allow, Mask, Confirm, Review, or Block. Prompts that are simply allowed or sanitized are sent on to an offline Ollama model. The system is to be evaluated using a synthetic labelled prompt dataset according to the predefined acceptance criteria.

Keywords: Prompt Screening, Sensitive Data Leakage Prevention, Prompt Injection Detection, Local Large Language Models

I. INTRODUCTION

This falls within the field of enterprise information security, more precisely concerning the governance of generative artificial intelligence (AI). Large language models (LLMs) are being used every day by students, developers, and employees for preparing documents, summarising information, writing code, and solving technical problems.

The particular advantage lies in altering the way data leaves an organisation: rather than going through validated form fields, the information is now typed directly into free-text prompts and sent on to the model, with only limited intermediate checking. The OWASP GenAI Security Project currently regards Prompt Injection (LLM01) and Sensitive Information Disclosure (LLM02) as the two most severe risks connected with LLM applications [15].

The paper deals with prompt screening, a control mechanism which, prior to inference taking place, checks each prompt, identifies any sensitive data entities and prompt risk indicators, and then applies a transparent policy.

The Enterprise AI Prompt Screening System examines the user, standardises the prompt, uses deterministic Java recognisers, calculates an explainable Prompt Risk Score, and then chooses one of five actions: Allow, Mask and Forward, Confirm, Human Review, or Block. Prompts that are allowed or have been sanitised are forwarded to a locally installed Ollama model via Spring AI, which allows the entire process to operate offline once the initial setup has been completed.

The system has six operational stages: those relating to users and roles, the web interface and intake, the screening layer, policy and masking, the screening decision, and controlled model access, all of these stages being supported by a privacy-aware audit repository. The prompt is never forwarded directly to the model; instead, each request is first evaluated and, if

needed, sanitised or rejected.

The study is based on three observations. The first is that conventional DLP is not designed to handle short, conversational prompts which combine legitimate task information with accidental secrets. The second point is that even though using the model locally eliminates the need for external providers, it does not protect the model from the kind of material that users send to it.

Thirdly, when a different AI model is used for carrying out moderation the decisions reached become hard to reproduce and audit, whereas risk management guidance such as the NIST AI Risk Management Framework stresses the importance of traceability and human oversight [16].

This paper therefore makes the following contributions: (i) a deterministic and model-independent technique for screening, (ii) an explainable weighted risk score that is linked to policy actions, and (iii) a measurable evaluation framework for the detection, masking and verification of sensitive data.

II. BACKGROUND AND DEFINITIONS

The method used for protecting sensitive data when it is in textual form has been around for a longer time than generative AI. Traditional enterprise data loss prevention (DLP) tools operated by using signatures and patterns on documents and on outgoing messages, but in the case of unstructured text the sensitivity usually resides in the context rather than in any one keyword [1]. As a result, research in the area of natural language processing gave up on relying on dictionaries and pattern matching and began to use named entity recognition and neural anonymisation models, since it became evident that no single method could ensure privacy [2]. Finally, conversational AI introduced a new way in which information could be leaked, because the conversations that users have with LLM services can themselves be made available or reconstructed [3].

The model's behaviour or causing hidden prompts to be leaked [4], and further research extended the attack to include instructions that were hidden within the content obtained from external sources [5].

The Prompt Risk Score (PRS) is a score going from 0 to 100 and is calculated by considering sensitive data exposure, the strength of the prompt injection signal, secret criticality, policy context, and obfuscation and repetition. It is linked to four risk categories—Low, Moderate, High, and Critical—and each of these categories has a default policy action.

III. RELATED WORK

Recursive neural networks have been used in the detection of sensitive information in unstructured documents in situations where sensitivity is context-dependent [1], and surveys of text anonymisation have compared NLP-based methods with those used in privacy-preserving data publishing [2]. In the process of carrying out text sanitisation, neural methods decide which sections to mask by referring to specific privacy risk measures [7], and pseudonymisation techniques have also been assessed in terms of their effect on privacy and on subsequent utility [9].

The studies deal with individual documents or are carried out using offline datasets; they highlight the poor assessment of real privacy risks, are based on models and demand a great deal of computing power, and are not connected with real-time prompt screening or policy decisions.

It has been found that personal information can leak out during conversations, even when such information is disclosed implicitly, thanks to learned classifiers that have been trained on collections of dialogue data [6], and it has already been shown that previous conversations with GPT models can be reconstructed in order to reveal private user content [3]. Furthermore, a synthetic dataset created by an LLM has also been made available for the purpose of detecting personally identifiable information (PII) [13].

The problem is that these methods offer only a limited degree of explainability and no entity-level masking, permit study leakage after submission rather than stopping sensitive data from entering the prompt, and take into account only the coverage of personal identifiable information and therefore omit credentials, organisational secrets, and cases involving prompt injection.

Research into prompt injection has demonstrated that it is possible to achieve goal hijacking and prompt leaking by using specially designed override instructions [4], that indirect prompt injection can be caused by data acquired through LLM-based applications [5], and that black box attacks have been carried out against real LLM-based applications [8]. At present, formal benchmarks systematically compare attacks with defences [10] and examine how instruction-tuned LLMs adhere to the instructions injected into the content [11].

The study focuses on carrying out attacks or on assessing the model's robustness rather than on considering external screening measures; it states that no single defence is adequate, and a principled evaluation shows that the defences mentioned are weaker than those asserted [12], thereby reinforcing the need for layered and auditable controls.

As shown in Table 1, the present research can be divided into two areas which are mostly independent of each other. The research dealing with sensitive data is able to detect or anonymise personal information from documents and conversations, but it does so offline and seldom puts its decisions into effect in real time. The research on prompt injection involves identifying the attacks and evaluating the defences at the model level, even though no existing method is capable of reliably preventing injection.

The usual disadvantages include the absence of a single control mechanism before inference, limited explainability, and poor auditability; our proposed system overcomes these weaknesses by incorporating deterministic detection, injection signals, and explainable policy decisions into a single gateway that is used locally.

Table 1. Summary of Reviewed Literature

Ref No	Author and Year	Method Used	Key Findings	Limitations
1	J. Neerbek et al. (2017)	Recursive neural networks	Detect sensitive information in unstructured documents where sensitivity depends on context	Document level classification; not designed for short prompts or for enforcing decisions before inference
2	P. Lison et al. (2021)	Survey of text anonymisation	Contrasts NLP based and privacy preserving data publishing approaches	Highlights weak evaluation of real privacy risk; offers no runtime control for LLM interaction
3	J. Chu et al. (2024)	Reconstruction of previous conversations with GPT models	Previous conversations can be reconstructed, exposing private user content	Studies leakage after submission; does not stop sensitive data from entering the prompt
4	F. Perez and I. Ribeiro (2022)	Handcrafted override instructions	Demonstrates goal hijacking and prompt leaking	Attack oriented; provides no deployable detection or policy layer
5	K. Greshake et al. (2023)	Indirect prompt injection	Introduces indirect prompt injection through data retrieved by LLM integrated applications	Maps the threat landscape; practical, auditable defences remain open
6	Q. Xu et al. (2020)	Personal information leakage detection in dialogue	Detects leakage including implicitly disclosed attributes	Learned classifiers trained on dialogue corpora; limited explainability and no entity level masking
7	A. Papadopoulou et al. (2022)	Neural text sanitisation	Decides what to mask using explicit privacy risk measures	Model based and compute intensive; not targeted at real time enterprise prompt gateways
8	Y. Liu et al. (2023)	Black box prompt injection attack	Evaluated against real LLM integrated applications	Confirms practical vulnerability but proposes no input side policy enforcement

Ref No	Author and Year	Method Used	Key Findings	Limitations
9	O. Yermilov et al. (2023)	Comparison of pseudonymisation techniques	Effect of pseudonymisation on privacy and downstream utility	Evaluated on offline datasets; not connected to real time prompt screening or policy decisions
10	Y. Liu et al. (2024)	Formal framework and benchmark	Systematically compares prompt injection attacks and defences	Finds no single defence sufficient; does not address sensitive data leakage
11	Z. Li et al. (2024)	Benchmark of instruction tuned LLMs	Measures how instruction tuned LLMs follow instructions injected into content	Evaluates model robustness, not external screening controls
12	Y. Jia et al. (2025)	Principled evaluation of defences	Evaluates defences on effectiveness and utility against adaptive attacks	Shows reported defences are weaker than claimed; supports layered, auditable controls
13	M. Savkin et al. (2025)	Synthetic, LLM generated dataset	Dataset for personally identifiable information (PII) detection	Covers PII only; excludes credentials, organisational secrets, and prompt injection cases

IV. PROPOSED ARCHITECTURE

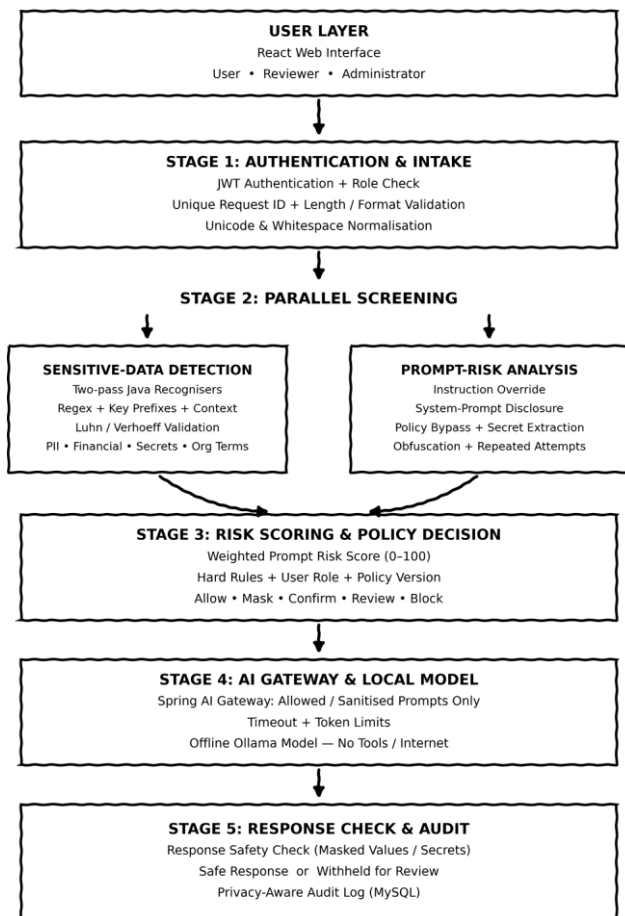


Figure 1. Proposed prompt screening pipeline of the Enterprise AI Prompt Screening System.

The backend of the system is a modular monolith developed using Spring Boot and Spring Security, together with a React single page frontend and a MySQL audit store; access to the models is achieved through Spring AI's provider-neutral chat abstraction so that the local Ollama model can be called [17]. Security is given the highest priority and the system is built in a step-by-step manner. The core pipeline—which includes all the stages of authentication, intake, detection, masking, policy decision, and audit logging—makes up the minimum viable system, while the dashboards, the response screening, and the reviewer workflow are only introduced later once the system has become stable. The fundamental concept is that the screening decision is deterministic and is made before the

model is invoked, thus ensuring that neither blocked nor unmasked prompts can reach the model at any point during the execution process. Figure 1 shows the screening pipeline.

1.1 Authentication and Prompt Intake

The Authentication and Access Control module deals with logins, is responsible for issuing JWTs, and handles roles (User, Reviewer, Administrator). The Prompt Intake and Normalisation module performs validation concerning length and format, applies Unicode and whitespace normalisation, assigns a unique request ID, and identifies any suspicious encodings.

All screening services are testable without a running model, so detection, masking, and policy behaviour can be verified independently of Ollama.

1.2 Sensitive Data Detection Engine

The process for detection has two stages: detection and anonymisation, a technique that was promoted by Microsoft Presidio [18] and has since been reimplemented as Java recognisers. In Pass 1, potential matches are identified using regular expressions, key prefixes, and keyword context. In Pass 2, these are checked; for cases involving payment cards the Luhn checksum must be passed, Aadhaar-type identifiers are verified using the Verhoeff check digit, and phone numbers are subject to length and context checks.

For each detection the following details are recorded: the entity type, the character span, the degree of confidence, and the suggested action; when there are overlapping matches, priority rules are employed to deal with the situation (for example, a match involving a private key is given higher priority than a match for a long text in general). However, according to the OWASP logging guidelines [19], only the entity categories and the number of entities are logged.

The first version of the recogniser catalogue contains email addresses (PERSONAL_EMAIL, [EMAIL]), phone numbers (PHONE_NUMBER, [PHONE_NUMBER]), PAN-type identifiers which are validated only through the use of a synthetic format rule and not by means of identity verification (PAN_LIKE_ID, [GOVT_ID_REMOVED]), 12-digit identifiers similar to Aadhaar with Verhoeff check digit validation (AADHAAR_LIKE_ID, [GOVT_ID_REMOVED]), payment cards that are recognised by means of a length pattern and a Luhn checksum (PAYMENT_CARD, [PAYMENT_CARD]), API keys and tokens that are detected using known prefixes, key/value

context and entropy heuristics (API_KEY, [SECRET_REMOVED]), passwords and secrets that are identified through keyword context and assignment patterns (PASSWORD_SECRET, [SECRET_REMOVED]), private key blocks that are recognised by PEM style header and footer patterns (PRIVATE_KEY, [PRIVATE_KEY_REMOVED]), and organisation terms that are derived either from an administrator's dictionary or from a project code pattern (ORG_CONFIDENTIAL, [INTERNAL_PROJECT]). Secret placeholders are applied only when a reviewer approves a sanitised version of a prompt; secrets are never automatically masked and forwarded.

1.3 Prompt Risk Analysis Engine

Since there is no rule-based or model-based detector capable of entirely dealing with the problem of prompt injection [10], the system intentionally focuses on the direct, user-visible signals mentioned in the OWASP guidance [15], [20]. Each signal detected is assigned an identifier, a weight, and an evidence span so that multiple weak signals can be combined instead of responding to a single keyword.

The types of risk involve instruction override (that is, an attempt to ignore, replace, or cancel trusted system instructions), system prompt disclosure (which occurs when someone asks that the prompts be printed, revealed, or reconstructed), policy bypass (requests to disable safety measures or to carry out actions outside the user's role), secret extraction (requests for credentials, tokens, or internal configuration information), suspicious role play (the use of role framing to justify prohibited disclosure or action), obfuscation signal (text that is encoded, fragmented, or heavily transformed and may be used to conceal instructions), and repeated attack pattern (multiple risk phrases or a number of blocked attempts that occur within a time window).

1.4 Prompt Risk Score and Policy Decision

Findings from both engines are combined into a weighted Prompt Risk Score,

$$PRS = 100 \times (0.35 F_1 + 0.25 F_2 + 0.20 F_3 + 0.10 F_4 + 0.10 F_5) \quad (1)$$

where each factor lies in [0, 1]: F_1 is sensitive data exposure (type and count of entities), F_2 is prompt injection signal strength, F_3 is secret criticality (rising sharply for passwords, keys, tokens, and private keys), F_4 is policy context (user role and policy version), and F_5 is obfuscation and repetition. Because the weights sum to 1, the score lies between 0 and 100.

The score is divided into four categories: if it is Low (0 to 24) the prompt is allowed through and normal usage metadata is recorded; if it is Moderate (25 to 49) the detected values are masked or a request is made to the user for confirmation; in the case of High (50 to 74) explicit confirmation or human review is required; and when the score is Critical (75 to 100) the prompt is blocked before it can reach the model.

Hard rules take precedence over score considerations since a private key block is always enforced, and an API key or password is either blocked or referred for mandatory review regardless of the situation; cases that are borderline are sent for human review rather than being resolved automatically, in accordance with the NIST AI RMF [16].

1.5 Masking Service

So long as the policy allows, whenever values have been detected they are substituted with typed placeholders rather than being deleted, since this enables the request to remain intelligible to the model and ensures that the response remains useful [9].

For example, if a prompt requests that the model analyses a sales report and then sends the summary to an email address, the email address, phone number and card number are replaced by [EMAIL], [PHONE_NUMBER] and [PAYMENT_CARD] before the modified prompt is given to the model, and the audit log includes only the entity counts, risk band and decision, without listing the original values.

1.6 AI Gateway and Response Safety Check

The gateway will only pass on those prompts which are allowed or have been sanitised, will correct the model identity, enforce the timeout and response length limits, and will not give the model access to any tools, the file system, or the internet, so that the possible impact of any injection which manages to escape screening is kept to a minimum.

The Response Safety Check will not send a response back until it has been displayed, and it will do so only if the response neither reproduces a masked value nor matches a configured secret pattern, forwarding it for review in such a case.

1.7 Privacy-Aware Audit Repository

The Audit Repository includes the entity categories, the counts, the risk band, the decision and the timing; it does not store any raw sensitive values. At each stage, privacy-aware audit logging is carried out and this involves the request ID, the categories, the counts, the risk band, the decision and the timing.

It helps achieve the system's governance objectives by offering measures to prevent data leakage, protecting against prompt injection, ensuring compliance with policy, monitoring usage, providing a human review workflow, and maintaining a privacy-aware audit trail, along with dashboards, alerts and reports for administrators.

V. METHODOLOGY

Step 1: An authenticated user enters a text prompt in the web interface.

Step 2: Authenticate and admit the prompt through a JWT and role check, assign a unique request ID, and apply length and format validation and Unicode and whitespace normalisation.

Step 3: Detect sensitive data with two-pass Java recognisers that find candidates and then confirm them with checksum and context, covering PII, financial data, credentials and organisation data.

Step 4: Analyse prompt risk for instruction override, system-prompt disclosure, policy bypass and secret extraction, and obfuscation and repeated attempts.

Step 5: Compute the weighted Prompt Risk Score from 0 to 100, reported with the factors that contributed to it.

Step 6: Combine the score, hard rules, user role and active policy version into one action with a plain-language reason: Block, Human review, Mask or confirm, or allow.

Step 7: Forward only allowed or sanitised prompts through the Spring AI gateway, applying timeout and token limits.

Step 8: Generate the response offline with the local Ollama model, with no tool or internet access.

Step 9: Scan the response for masked values, secret patterns and blocked phrases; return a safe response to the user, or withhold the response and flag it for review.

VI. IMPLEMENTATION PLAN AND EVALUATION

The architecture, pipeline, policy framework, masking method, gateway design, and evaluation criteria have already been set, but the work on implementation and integration is still being carried out. The results referred to below are therefore expected outcomes based on the design; they have not been experimentally verified and include no figures relating to accuracy, precision, recall, or latency. They will be confirmed or modified following testing on the synthetic dataset.

Since there is no publicly available benchmark for this prompt screening scenario, the evaluation will be carried out by means of a specially designed synthetic labelled dataset, using the same method as that employed in the case of the synthetic PII corpora [13]. The dataset will include safe prompts, prompts involving sensitive data relating to either single or multiple categories, direct injection examples based on published attack patterns [4], obfuscated variants, and negative controls; moreover, no actual personal or production data will be used.

The criteria are entity precision (that is, the number of correctly detected entities divided by the total number of detections), entity recall (that is, the number of correctly detected entities divided by the total number of entities that were expected), the F1 score (which is the harmonic mean of precision and recall [14]), masking completeness (the number of values that have been correctly replaced divided by the number of values that were chosen for masking), policy accuracy (the number of scenarios which result in the decision that has been configured divided by all the scenarios), log privacy (the review of the audit records for any raw sensitive values), screening latency (the average and maximum screening times on the declared hardware), and reproducibility (the ability to get the same results when the versions of the rules and policy are the same).

The conditions for acceptance are that all the categories listed for the target recogniser must be identified, together with the precision, recall, and F1 score associated with each category; each value used for masking must be substituted by the appropriate placeholder before it is sent on; seeded direct risk prompts must increase the expected signal without causing a reaction to harmless prompts; and each controlled scenario must result in the action which has been set up for it (that is, Allow, Mask, Confirm, Review, or Block).

Furthermore, users cannot obtain other users' prompts, reviews, or administrative data by altering their requests; the audit records include only the categories, the counts, and the decision metadata, not the original values; after it has been set up, the entire workflow, comprising the Ollama inference process, works off the internet; for all decisions except 'Allow', the matched category, the reason according to the policy, and a corrective action are shown; and if the same input is submitted under the same rules and policy, the same decision is returned.

As with payment cards and Aadhaar identifiers, which are checked using a checksum, greater accuracy can be expected compared to context-dependent items such as passwords and

organisation names since the verification process eliminates random chains of digits that happen to fit a certain format. With regard to prompt risk, the system is expected to detect explicit and directly stated attacks but will not be able to identify paraphrased, heavily obfuscated, or indirect injections, this being in line with the evidence that current defences are still incomplete [12].

Screening latency is expected to be much lower than that of local model inference since the detection method involves pattern matching and checksum calculation instead of using an extra model; this expectation will be verified by separately measuring the times for screening and for inference. The decisions are also expected to be entirely reproducible as the pipeline includes no stochastic components.

A script, which can be repeated, will be used to carry out end-to-end behaviour verification by submitting a safe prompt, a prompt containing synthetic personal data, a prompt with a synthetic API key, an example involving direct injection; causing a high risk item to be escalated via the Reviewer role; and having the Administrator confirm that the audit record contains only masked metadata.

As long as the criteria are met, organizations should establish a technically enforced barrier to prevent personal, financial and credential data from being accidentally made public; they should keep an auditable record of AI usage which does not contain any of the raw sensitive values; their policy enforcement system should know the user's role; and they must be able to use generative AI offline rather than relying on a commercial cloud provider. The expected limitations are that the coverage will be limited to that provided by the recognizer catalogue, that false positives will happen when legitimate prompts resemble sensitive formats, and that no protection claim can be made against indirect or highly obfuscated injection.

VII. CONCLUSION AND FUTURE SCOPE

The paper outlines an Enterprise AI Prompt Screening System which includes a deterministic and explainable control mechanism situated between enterprise users and a locally installed language model. The design incorporates rule-based recognisers for sensitive data, detectors for direct prompt injection, a weighted Prompt Risk Score, and five policy actions, together with typed placeholder masking, a restricted Spring AI gateway, response safety verification, and privacy-aware audit logging. The main contribution of the system is the separation of the security decision from the language model, thereby ensuring that all the results are reproducible, auditable, and independent of the model's behaviour.

Since the implementation is still being carried out, the paper only offers a reference architecture and a method for evaluation based on measurable criteria rather than any empirical results. In future work the results will be provided and the system will be extended to include file and image inputs, multilingual detection, integration with enterprise identity systems, support for multiple approved models, and the ability to detect indirect prompt injection in retrieval augmented generation.

VIII. REFERENCES

- [1] J. Neerbek, I. Assent, and P. Dolog, "TABOO: Detecting Unstructured Sensitive Information Using Recursive Neural Networks," in Proc. IEEE 33rd International Conference on Data Engineering (ICDE), 2017, pp. 1399-1400.

- [2] P. Lison, I. Pilán, D. Sánchez, M. Batet, and L. Øvrelid, "Anonymisation Models for Text Data: State of the Art, Challenges and Future Directions," in Proc. 59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing, 2021, pp. 4188-4203.
- [3] J. Chu, Z. Sha, M. Backes, and Y. Zhang, "Reconstruct Your Previous Conversations! Comprehensively Investigating Privacy Leakage Risks in Conversations with GPT Models," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2024, pp. 6584-6600.
- [4] F. Perez and I. Ribeiro, "Ignore Previous Prompt: Attack Techniques for Language Models," in Proc. NeurIPS ML Safety Workshop, 2022, arXiv:2211.09527.
- [5] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection," in Proc. 16th ACM Workshop on Artificial Intelligence and Security (AISEC), 2023.
- [6] Q. Xu, L. Qu, Z. Gao, and G. Haffari, "Personal Information Leakage Detection in Conversations," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6567-6580.
- [7] A. Papadopoulou, Y. Yu, P. Lison, and L. Øvrelid, "Neural Text Sanitization with Explicit Measures of Privacy Risk," in Proc. 2nd Conference of the Asia-Pacific Chapter of the ACL and 12th International Joint Conference on Natural Language Processing, 2022, pp. 217-229.
- [8] Y. Liu et al., "Prompt Injection Attack Against LLM-Integrated Applications," arXiv preprint arXiv:2306.05499, 2023.
- [9] O. Yermilov, V. Raheja, and A. Chernodub, "Privacy- and Utility-Preserving NLP with Anonymized Data: A Case Study of Pseudonymization," in Proc. 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP), 2023, pp. 232-241.
- [10] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and Benchmarking Prompt Injection Attacks and Defenses," in Proc. 33rd USENIX Security Symposium, 2024, pp. 1831-1847.
- [11] Z. Li, B. Peng, P. He, and X. Yan, "Evaluating the Instruction-Following Robustness of Large Language Models to Prompt Injection," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2024, pp. 557-568.
- [12] Y. Jia, Z. Shao, Y. Liu, J. Jia, D. Song, and N. Z. Gong, "A Critical Evaluation of Defenses Against Prompt Injection Attacks," arXiv preprint arXiv:2505.18333, 2025.
- [13] M. Savkin, T. Ionov, and V. Konovalov, "SPY: Enhancing Privacy with Synthetic PII Detection Dataset," in Proc. 2025 Conference of the NAACL: Human Language Technologies (Student Research Workshop), 2025, pp. 236-246.
- [14] Y. Sasaki, "The Truth of the F-Measure," Teach Tutor Mater, 2007.
- [15] OWASP GenAI Security Project, "OWASP Top 10 for LLMs and Generative AI Applications," 2025.
- [16] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023.
- [17] Spring, "Spring AI Reference Documentation: Chat Client API and Ollama Chat," Spring Documentation, 2026.
- [18] Microsoft, "Presidio: Data Protection and De-Identification SDK," Microsoft Open Source Documentation.
- [19] OWASP Cheat Sheet Series, "Logging Cheat Sheet," OWASP Foundation.
- [20] OWASP Cheat Sheet Series, "LLM Prompt Injection Prevention Cheat Sheet," OWASP Foundation.
- [21] K. Boeckl and N. Lefkowitz, "NIST Privacy Framework: A Tool for Improving Privacy Through Enterprise Risk Management, Version 1.0," NIST Cybersecurity White Paper, 2020.
- [22] R. Geng, C. Yin, Y. Wang, Y. Chen, and J. Jia, "PIArena: A Platform for Prompt Injection Evaluation," in Proc. 64th Annual Meeting of the Association for Computational Linguistics, 2026, pp. 33170-33192.
- [23] G. Loiseau, D. Sileo, D. Riquet, M. Meyer, and M. Tommasi, "Tau-Eval: A Unified Evaluation Framework for Useful and Private Text Anonymization," in Proc. EMNLP 2025: System Demonstrations, 2025, pp. 216-227.
- [24] Y. Ren, K. Ramesh, Y. Yao, and A. Field, "How Do We Measure Privacy in Text? A Survey of Text Anonymization Metrics," in Findings of IJCNLP-AAACL, 2025, pp. 1532-1544.
- [25] N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz, and S. Zanella-Béguelin, "Analyzing Leakage of Personally Identifiable Information in Language Models," in Proc. IEEE Symposium on Security and Privacy (SP), 2023.
- [26] I. Pilán, P. Lison, L. Øvrelid, A. Papadopoulou, D. Sánchez, and M. Batet, "The Text Anonymization Benchmark (TAB): A Dedicated Corpus and Evaluation Framework for Text Anonymization," Computational Linguistics, vol. 48, no. 4, pp. 1053-1101, 2022.
- [27] J. Yi et al., "Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models," arXiv preprint arXiv:2312.14197, 2023.
- [28] K. Hines, G. Lopez, M. Hall, F. Zarfati, Y. Zunger, and E. Kiciman, "Defending Against Indirect Prompt Injection Attacks with Spotlighting," arXiv preprint arXiv:2403.14720, 2024.
- [29] S. Chen, J. Piet, C. Sitawarin, and D. Wagner, "StruQ: Defending Against Prompt Injection with Structured Queries," in Proc. 34th USENIX Security Symposium, 2025.
- [30] E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, and A.

Beutel, “The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions,” arXiv preprint arXiv:2404.13208, 2024.

[31] H. Inan et al., “Llama Guard: LLM-Based Input-Output Safeguard for Human-AI Conversations,” arXiv preprint arXiv:2312.06674, 2023.

[32] T. Rebedea, R. Dinu, M. Sreedhar, C. Parisien, and J. Cohen, “NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails,” in Proc. EMNLP 2023: System Demonstrations, 2023.

[33] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT),” IETF RFC 7519, May 2015.

[34] Government of India, “The Digital Personal Data Protection Act, 2023,” Act No. 22 of 2023.

[35] OWASP GenAI Security Project, “LLM01:2025 Prompt Injection,” OWASP Foundation, 2025.

[36] OWASP GenAI Security Project, “LLM02:2025 Sensitive Information Disclosure,” OWASP Foundation, 2025.

[37] OWASP Cheat Sheet Series, “Authorization Cheat Sheet,” OWASP Foundation.

[38] A. Wilkinson, “Spring Boot 4.1.0 Available Now,” Spring Blog, Jun. 10, 2026.

[39] Oracle, “MySQL 8.4 Reference Manual,” Oracle Corporation.

[40] Meta Open Source, “React Documentation and React 19 Reference,” react.dev.