IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

Architectural Vulnerabilities, Privacy Leakage, and Defense Mechanisms in Large Language Model Deployments: A Comprehensive Survey

Prof. Sangita K. Chaudhari[1], Prof. Mahendra S. Dalvi[2], Himani Kishor Mane[3], Snehal Nandakishor Gavas[4], Payal Raghunath Katore[5], Prajakta Dipak Patil[6]
Head of Department Information Technology[1]
Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India
Savitribai Phule Pune University (SPPU)
sangita.chaudhari@sitrc.org, dalvimahendra21@gmail.com, himanimane2205@gmail.com
gavassnehal4@gmail.com, payalkatore2005@gmail.com, sp2844273@gmail.com

Abstract: *The rapid integration of Large Language Models (LLMs) into enterprise systems has exposed key architectural vulnerabilities resulting from the inability to strictly separate untrusted user data channels from control instructions. Adversarial exploits such as direct and indirect prompt injections, jailbreaks, and sensitive data exposure represent major security vectors. Traditional Web Application Firewalls (WAFs) fail to parse complex semantic obfuscation, whereas secondary LLM evaluators introduce untenable runtime latency (¿400ms).*

This paper presents a structured review of state-of-the-art security, privacy, and sanitization guardrails in LLM architectures. By evaluating existing frameworks across latency, detection strategy, PII redaction capabilities, and continuous retrain adaptability, we identify key operational gaps. Furthermore, we summarize architectural requirements for low-latency, hybrid security middleware models capable of operating under sub-25ms response boundaries..

Keywords: Large Language Models, Prompt Injection, PII Redaction, Machine Learning, Defense Middleware, LLM Security, Data Privacy.

I. INTRODUCTION

The rapid deployment of Large Language Models (LLMs) across corporate ecosystems has redefined interactive software architectures [1]. Applications now leverage generative models for natural language database querying, autonomous agency, automated code synthesis, and enterprise context summarization [1].

However, because LLMs operate on probabilistic context continuation rather than deterministic logic execution, they lack fundamental boundaries between core system instructions and untrusted user inputs [1]. Adversarial entities exploit this fundamental boundary confusion through prompt injection attacks, wherein engineered prompts bypass system safety alignment, override operational rules, trigger unauthorized functions, or leak sensitive system prompts [1].

Simultaneously, unmonitored user interactions present severe corporate data compliance risks due to the accidental exposure of Personally Identifiable Information (PII) to commercial LLM provider APIs [1]. Enterprise LLM safety encounters a fundamental trade-off between threat detection precision and operational execution speed:

1) Failure of Static Rules: Web Application Firewalls (WAFs) relying on static blocklists and keyword matches are easily bypassed by Base64 encoding, character substitution, leetspeak, or roleplay framing (e.g., ”DAN” jailbreaks) [1].

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-40336

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

2) High Latency of Secondary LLM Guards: Deploying secondary evaluator LLMs (e.g., Llama Guard) adds hundreds of milliseconds of runtime overhead per request, causing unacceptable performance bottlenecks in high-throughput enterprise APIs [1]

3) Decoupled Security and Privacy: Existing safety tools often address either prompt injection classification or structural parsing, while ignoring automated PII redaction and privacy preservation [3], [5]

This review paper categorizes threat paradigms, systematically surveys recent academic literature, analyzes critical gaps in state-of-the-art defenses, and presents formal mathematical and architectural specifications for real-time security middleware proxy frameworks.

II. LITERATURE SURVEY

II.1 Related Research

Recent research in AI safety highlights the operational limitations of static and heavy-model defenses when applied to generative environments. Wang et al. [1] conducted a systematic review of software vulnerability analysis using LLMs, highlighting that while LLMs aid software analysis, their probabilistic architecture introduces complex attack vectors. De Freitas et al. [2] explored context vulnerabilities and self-disclosure in generative AI chatbots, showing how natural conversations can be manipulated to extract unintended information. Sperl ı et al. [3] evaluated privacy risks, highlighting that partial privacy scanners leave endpoints vulnerable to inference attacks. Nikhil et al. [4] evaluated LLM security in penetration testing, showing that while deep logical flaw detection is effective, it incurs severe computational latency exceeding 400ms. Pervez et al. [5]

Proposed an autonomous agent defense utilizing prompt sanitization that achieved low processing delays (< 20ms), but lacked continuous model retraining loops to protect against evolving exploits. Wang et al. [6] established an attack taxonomy classifying prompt injection vectors by objective. Liu et al. [7] introduced a benchmarking framework for formal prompt injection defenses. Chen et al. [8] proposed StruQ, utilizing structured queries to separate control and data channels at sub-15ms execution speeds. However, Li et al. [9] demonstrated that classifier-based defenses often overfit to superficial surface patterns rather than capturing true semantic intent. Finally, Chang et al. [10] demonstrated indirect prompt injections in Retrieval-Augmented Generation (RAG) pipelines.

II.2 Comparative Gap Analysis

Table I compares existing security approaches across core technical metrics: methodology, primary detection strategy, operational latency, PII masking support, and offline retrain adaptability.

Analysis of the state-of-the-art reveals three primary gaps:

1) Absence of Continuous Model Retraining: Current systems lack automated pipelines to persist newly discovered zero-day injection payloads into isolated threat stores for offline model fine-tuning.

2) Isolation of Security and Privacy Mechanisms: Systems generally prioritize injection classification or query parsing, neglecting unified, real-time entity redaction (Regex + NER).

3) Latency Overhead: : High-fidelity semantic inspection frameworks often introduce latencies that exceed modern application performance budgets (> 100ms).

III. METHODOLOGY AND SYSTEM MODEL

To resolve these architectural limitations, modern security middleware proxies combine lightweight feature extraction, dual-stage PII sanitization, classical machine learning classification, and threat logging.

A. Pipeline Architecture:-

The processing sequence consists of five stages:

1) Text Normalization: Stripping non-printable control characters and executing Unicode NFKC normalization.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, October 2026 Impact Factor: 8.2

2) PII Masking: Redacting sensitive entities (e.g., SSNs, credit cards, emails) using combined Regular Expressions and Named Entity Recognition (NER).
3) Feature Vectorization: Mapping sanitized text into dense numeric embeddings using TF-IDF and transformer-based sentence encoders.
4) Threat Evaluation: Computing threat probability scores using lightweight tree-based or support-vector estimators (e.g., XGBoost, SVM).
5) Action Execution: Blocking malicious inputs and persisting vectors to a threat store (DBthreats), or forwarding sanitized text to target LLM API endpoints.sssl

B. Mathematical Model:-

Let P represent the raw input text string such that $P \in \Sigma*$.

1. Text Normalization (N):

$$P' = N(P) \quad (1)$$

2. PII Sanitization (M): Combining Regex (R) and Named Entity Recognition (NER):

$$P\text{sanitized} = M(P') = NER(R(P')) \quad (2)$$

3. Feature Vector Mapping (E):

$$V = E(P\text{sanitized}) \in \mathrm{R}d \quad (3)$$

4. Threat Probability Score Evaluation ($f\theta$):

$$S = f\theta(V) \text{ where } S \in [0, 1] \quad (4)$$

5. Decision Logic (T): Given a predetermined security risk threshold $T \in (0, 1)$:

$$\text{Action}(P) = \begin{cases} \text{Block \& Log to } DB\text{threats, if } S > T \\ \text{Forward } P\text{sanitized to Target LLM, if } S < T \end{cases} \quad (5)$$

6. Operational Latency Constraint (Δt):

$$\Delta t\text{total} = tN + tM + tE + tf \leqslant 25 \text{ ms} \quad (6)$$

IV. SYSTEM DESIGN AND ARCHITECTURE

An enterprise-grade defense framework comprises three core operational layers:
A. Data Processing & Sanitization Layer Interprets incoming REST API payloads asynchronously via high-throughput ASGI services (e.g., FastAPI). Text normalization strips adversarial obfuscation symbols prior to dual-stage PII detection.
B. Adaptive Threat Inspection Engine Converts normalized text into dense feature vectors. Classical estimators (XGBoost/SVM)
evaluate vectors against trained classification boundaries. This approach avoids the computational overhead of heavy secondary LLMs, keeping total screening delays below 25ms.
C. Action Dispatch & Threat Logging Module Requests meeting or exceeding the risk threshold T are immediately dropped, returning a standardized security violation payload to the client. The corresponding query vectors and metadata are stored in
DBthreats, facilitating asynchronous offline retraining cycles to adapt against newly emerging jailbreak patterns. Safe prompts ($S < T$) are routed to target backend models.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

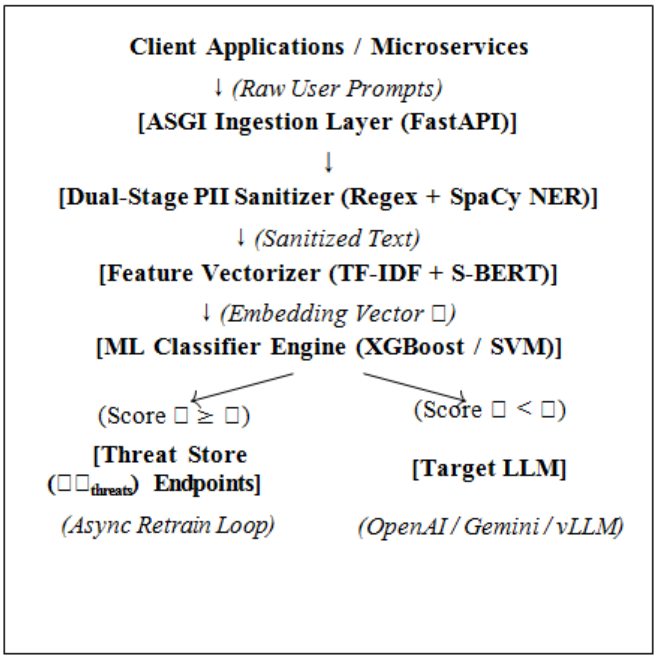

Figure 1 Architectural Block Diagram of Real-Time Security Proxy

Reference	Year	Methodology	Detection Strategy	Latency	PII Masking	Adaptive Retrain
Wang et al. [1]	2024	Vulnerability Review	Static Code Analysis	N/A	No	No
De Freitas et al. [2]	2024	Chatbot Disclosure	Self-Disclosure Mapping	N/A	No	No
Sperlí et al. [3]	2023	Risk Framework	Prompt/Inference Scanners	~50ms	Partial	No
Nikhil et al. [4]	2024	Pen-Testing Model	Logical Flaw Detection	>400ms	No	No
Pervez et al. [5]	2025	Agent Defense	Prompt Sanitization	<20ms	Full	No
Wang et al. [6]	2024	Attack Taxonomy	Objective Threat Mapping	N/A	No	No
Liu et al. [7]	2024	Formal Benchmark	Attack/Defense Evaluation	~50ms	No	No
Chen et al. [8]	2025	StruQ Framework	Control/Data Separation	<15ms	No	No
Li et al. [9]	2026	Surface Heuristics	Feature Overfitting Detection	125ms	No	No
Chang et al. [10]	2026	Indirect Shield	RAG Context Inspection	100ms	Partial	No

Table 1 Comparative Research Gap Matrix of Existing LLM Defenses

V. TECHNICAL INFRASTRUCTURE REQUIREMENTS

To maintain low execution latencies under high request concurrency, appropriate deployment infrastructure is required. Table II and Table III summarize core deployment recommendations.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

TABLE II: Infrastructure HARDWARE Specifications

Resource	Minimum Spec	Target Recommendation
CPU	Quad-Core i5/Ryzen 5	8-Core Intel i7/Xeon
RAM	8 GB DDR4	16 GB / 32 GB DDR4
Storage	20 GB SSD	100 GB NVMe SSD
Network	100 Mbps Ethernet	1 Gbps Full-Duplex

TABLE: III Software Technology Stack

Component Layer	Technology Selected
Operating System	Ubuntu Linux 22.04 LTS / Windows 11
Programming Language	Python 3.9+
API Proxy Framework	FastAPI (Uvicorn ASGI)
NLP & Entity Parsing	SpaCy / Hugging Face Transformers
Machine Learning	Scikit-learn / XGBoost
Threat Repository	PostgreSQL / MongoDB
Telemetry UI	React.js 18.0+

VI. APPLICATIONS AND FUTUREWORK

A. Applications

• Enterprise AI Portals: Guarding internal employee- facing conversational assistants from corporate data leaks.

• RAG Pipelines: Inspecting external knowledge inputs retrieved from unverified third-party databases to block indirect prompt injections.

• Multi-Tenant API Gateways: Serving as an inline security layer for software vendors deploying generative AI features.

B. Future Scope

• Multimodal Inspection: Expanding feature extraction pipelines to evaluate text embedded within images, audio, and video payloads.

• Decentralized Threat Intelligence: Implementing federated learning protocols across enterprise nodes to securely share anonymized threat vectors without exposing proprietary user data.

• Hardware Acceleration: Utilizing TensorRT and ONNX runtimes to further optimize vector embedding extraction, aiming for sub-10ms processing limits.

VII. CONCLUSION

This paper reviewed the evolving threat landscape of Large Language Model deployments, focusing on prompt injection vectors, jailbreak techniques, and privacy risks. By analyzing existing literature, key operational trade-offs between detection precision, privacy protection, and latency overhead were iden- tified. Lightweight, hybrid security proxies—combining dual- stage PII sanitization, vector embeddings, and classical ma- chine learning models—offer a viable solution. By maintaining sub-25ms response times and integrating continuous offline threat logging, such architectures provide scalable guardrails for enterprise generative AI deployments.

REFERENCES

[1] L. Wang, J. Li, Z. Li, S. Lu, and G. Yang, “Enhancing Software Vulnerability Analysis via Large Language Models: A Systematic Review,” in Proc. IEEE 9th Int. Conf. Data Sci. Cyberspace (DSC), IEEE, 2024.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

[2] J. De Freitas et al., "Disclosure, Humanizing, and Contextual Vulnerability of Generative AI Chatbots," Policy Paper, Harvard Business School Working Forum, 2024.
[3] G. Sperlí et al., "Exploring Privacy and Security Risks in LLMs: Data Leakage, Prompt Injection, and Membership Inference," in Proc. IEEE Int. Symp. Multimedia (ISM), IEEE, pp. 312–319, 2023.
[4] M. Nikhil, A. Ramnarayan, Abhinav U., and M. Lakshmi, "Evaluating and Exploiting Security Vulnerabilities in Large Language Models (LLMs) for Offensive Penetration Testing," in Proc. Int. Conf. Circuit Power Comput. Technol. (ICCPCT), IEEE, pp. 1045–1051, 2024.
[5] M. J. Pervez, A. Lekesiz, H. Tsang, and M. H. Bhuyan, "Securing LLM agents: From prompt sanitization to autonomous red teaming and beyond," Internet of Things and Cyber-Physical Systems, vol. 5, pp. 101–114, 2025.
[6] N. Wang, K. Walter, Y. Gao, A. Abuadabba, G. Stringhini, and V. Moonsamy, "Understanding the Adversarial Landscape of Large Language Models Through the Lens of Attack Objectives," IEEE Security & Privacy, vol. 22, no. 3, pp. 45–56, 2024.
[7] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and Benchmarking Prompt Injection Attacks and Defenses," in Proc. 33rd USENIX Secur. Symp. (USENIX Security '24), Philadelphia, PA, USA, pp. 1831–1847, 2024.
[8] S. Chen et al., "StruQ: Defending Against Prompt Injection with Structured Queries," in Proc. 34th USENIX Secur. Symp. (USENIX Security '25), Seattle, WA, USA, 2025.
[9] S. Li, C. Yu, Z. Ni, H. Li, C. Peris, C. Xiao, and Y. Zhao, "Defenses Against Prompt Attacks Learn Surface Heuristics," in Proc. 64th Annu. Meet. Assoc. Comput. Linguist. (ACL), San Diego, CA, USA, pp. 10970–10987, 2026.
[10] H. Chang, E. Bao, X. Luo, and T. Yu, "Overcoming the Retrieval Barrier: Indirect Prompt Injection in the Wild for LLM Systems," in Proc. 35th USENIX Secur. Symp. (USENIX Security '26), USENIX Association, 2026.