International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 4, July – August 2026

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

"LLM Jailbreak & Prompt Injection Firewall"

Shreyas G C
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
shreyasgcpatel09@gmail.com

Mr. Raghu M T
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
raghu.mt1@gmail.com

Monisha R
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
monisharamesh7078@gmail.com

Nishanth K L
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
nishanthgowda978@gmail.com

Punith M
Dept of ISE
Don Bosco Institute of Technology
Bengaluru, India
punithpuni8122004@gmail.com

----------**********----------

Abstract:

As large language models (LLMs) evolve from chatbots into integral parts of autonomous agents within enterprise ecosystems, attack surfaces grow exponentially, both in number and complexity. This paper provides an up-to-date survey of the landscape of LLM jailbreaking and prompt injection, with a specific focus on indirect and multimodal injection methods, which pose a significant threat to the safety alignment of LLMs. We investigate the practical efficacy of different LLM firewalls, which aim to offer runtime guardrails between the user querying the model and the model itself by performing input and output sanitization. We find that many existing approaches offer only a limited safety guarantee, due to the fact that most of them apply only one detection layer to classify potentially hazardous inputs. We propose a multi-layer sequential defense pipeline, which employs both light-weight pattern classification and semantic analysis based on an LLM, in which each layer is evaluated individually.

----------**********----------

I. INTRODUCTION

Moving artificial intelligence from simple chatbots into the heart of automated business systems has completely rewritten the rules of digital security. Today, in 2026, these AI programs almost never work alone. Instead, they are wired directly into corporate networks, fetching live internet data and triggering outside services to get their jobs done. However, this deep connection reveals a major structural weakness regarding how commands and normal information overlap, since regular software strictly separates the two. AI language models completely ignore this boundary. Whether it is a developer's strict rule, a customer's question, or a random website snippet, the system throws it all into one bucket and treats it exactly the same. That exact confusion between active commands and passive text is the root cause of modern AI hacking. Hackers can either trick the system into ignoring its safety training entirely, or they can sneak in hidden instructions that force the AI to execute unauthorized actions.

Unsurprisingly, the way attackers exploit these vulnerabilities has evolved quickly. In the beginning, people just used clever role-playing games or alternate personas to convince the chatbot to break its own rules. Now, the tactics are much more sophisticated and sneakier, often using automated attacks or planting hidden traps where the actual user is completely innocent. An AI assistant could easily get hijacked just by scanning a maliciously coded document or a rigged email while trying to help you with a standard request. This creates a massive headache for companies, as a hijacked bot might silently expose private files, wipe out databases, or send out scam messages on its own without anyone ever typing a bad command.

Since these threats are now happening in real life rather than just in theoretical labs, relying solely on the AI's initial safety training is no longer a viable defense strategy against persistent hackers. That realization has sparked the rise of specialized AI firewalls—often referred to as content guardrails—which are now essential tools for securing smart systems. Much like traditional security software scans incoming web traffic for viruses, these new barriers actively monitor every single word flowing into and out of the language model in real time.

II.RELATED WORK

Hagen Dorff et al. [1] looked into a fairly unsettling development: "autonomous jailbreak agents" built on Large Reasoning Models. They showed that models like DeepSeek-R1 and Grok-3 can be turned around and used as attackers against other LLMs, with success rates climbing as high as 97.14%. The work makes a strong case that better reasoning also means better attacking — but it's really about model-vs-model conflict, not the kind of stealthy, human-crafted injection this project is concerned with.

Liu et al. [2] took a genetic-algorithm approach, evolving "persona prompts" automatically until they reliably slipped past safety alignment. The result was a 70% drop in refusal rates, achieved mainly by burying harmful intent inside elaborate fictional characters. It's a solid demonstration of direct prompt injection, but the study never tests these personals against a runtime semantic firewall that's actually looking for rolepl5aybased evasion — which is a gap this project tries to close.

Zheng et al. [3] built a purification pipeline aimed squarely at indirect prompt injection in RAG pipelines, treating anything pulled from an external source as inherently untrustworthy and running it through a cleanup pass before it reaches the model. It works, but the extra processing stage adds enough latency that it's a hard sell for anything conversational or real-time.

Karanam et al. [4] reviewed the broader "Safety Stack" landscape, with a close look at Nemo Guardrails and Llama Guard. Both catch overt toxicity well enough, but the review points out a recurring problem: a conservative bias that produces a lot of false positives and gets in the way of normal, legitimate use. Their takeaway is that firewalls need sharper intent-recognition, not just blunter filtering.

Wang et al. [5] borrowed language from traditional cybersecurity to build the "Prompt ware Kill Chain," mapping LLM exploitation onto familiar phases — initial access, privilege escalation, exfiltration. It's a genuinely useful way to think about where defenders could place tripwires, but the paper stays at the strategic level and stops short of any actual working implementation.

Hu et al. [6] introduced Dual Breach, showing that an attacker can defeat both the model's internal alignment and an external firewall at the same time using a target-driven initialization strategy. The finding worth noting here is a "robustness gap" — the same prompt that jailbreaks the model can fool the firewall guarding it. The catch is that this requires expensive, high-dimensional optimization, which limits how practical the attack is outside a research setting.

Zou et al. [7] worked on gradient-based universal adversarial suffixes (GCG) — strings of near-gibberish text engineered to push a model toward a "Yes" response regardless of the question. This was an important early signal that keyword-based safety filters don't hold up well. On the flip side, these suffixes tend to have unusually high perplexity, which makes them fairly easy for a modern semantic firewall to flag.

Patel et al. [8] compared adversarial unlearning against runtime filtering and found that stripping toxic data out of the model's weights directly gives more durable protection than bolting a filter on afterward. The tradeoff is what they call an "alignment tax" — unlearning can quietly degrade the model's general reasoning along with the bad behavior.

Sharma et al. [9] built a real-time Semantic Context Analyzer that compares a user's actual intent against whatever context RAG retrieval pulls in, using embeddings to catch the mismatch. It does a good job surfacing hidden injections buried in scraped web content. Where it falls short is novel obfuscation — things like ASCII art or Base64 encoding — which can still slip through undetected.

III. PROPOSED SYSTEM

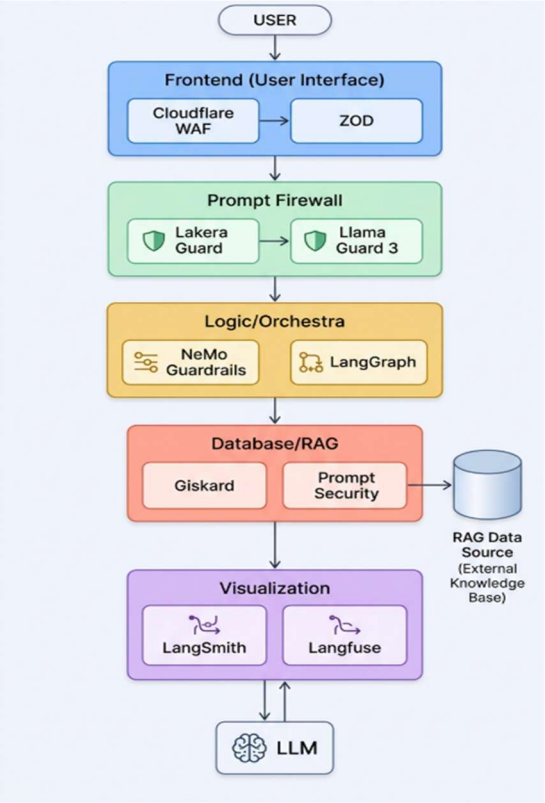

Fig 3.1 The proposed system

To safeguard execution, this multi-tier infrastructure handles incoming user requests across a structured sequential workflow prior to returning any final results. The process begins by harvesting safety metrics as web threats, broken payloads, and adversarial prompt attacks get blocked by Lakera Guard, Llama Guard 3, Cloudflare WAF, and ZOD. Following this, execution flow and complex reasoning paths are coordinated through Lang Graph alongside Nemo Guardrails, whereas validation tools including Prompt Security and Giscard evaluate information retrieved from enterprise knowledge stores. In the end, this verified background data reaches the core model to produce trustworthy outputs, supported by Lang fuse and Lang Smith to track real-time operational quality continuously.

IV.METHODOLOGY

Safeguard measures for generative AI systems commence by routing every incoming query through rigid static filters to clean raw content, excise documented attack signatures, flag obfuscated

character strings like Base64, and eliminate malicious token patterns. Subsequently, the query advances toward purpose evaluation and contextual understanding using specialized guardrail classifiers including Lakera Guard and Llama Guard along with embedding-based spatial comparison. This combined approach uncovers, personal-bypass tactics, and hidden instruction injection attacks concealed inside external background documents retrieved by the application.

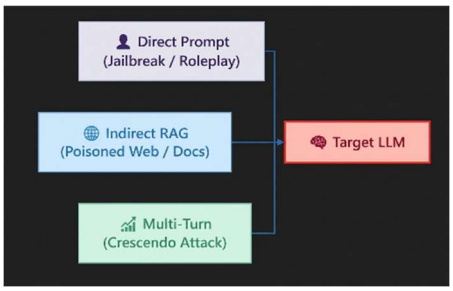

Fig 4.1 Attack Pathways

To prevent boundary crossing, the system enforces structural partition and runtime segregation by wrapping inputs in rigid JSON or XML data boundaries and employing a two-tiered processing setup. Strict delimiters and a dual-model setup isolate untrusted data before reaching the main engine.

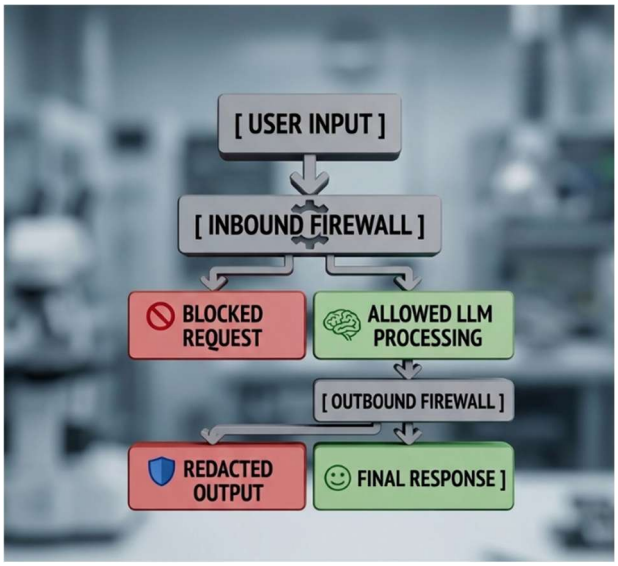

Fig 4.2 High-Level Methodology Overview

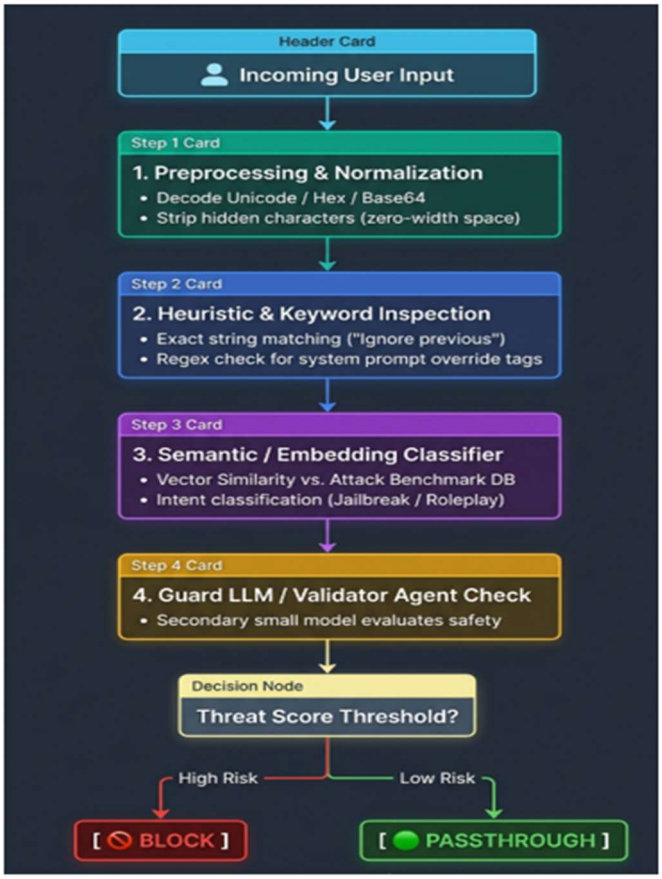

Fig.4.3 Inbound Prompt Injection Layer

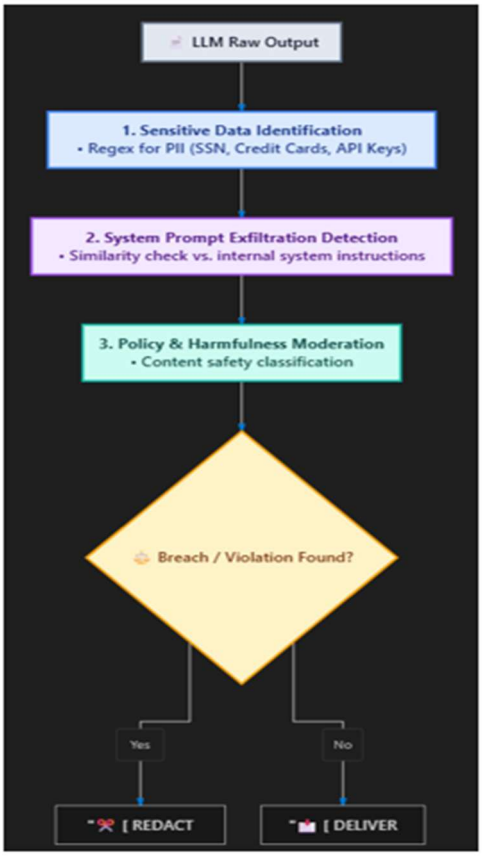

Fig 4.4 Outbound Response Validation Layer

V. SYSTEM FEATURES

Category	Features
Prompt Injection Defense	Direct and indirect injection detection, semantic intent analysis, base64/hex payload decoding, delimiter manipulation filtering, prompt boundary enforcement, adversarial prefix neutralization
Jailbreak Mitigation	Roleplay & DAN exploit blocking, adversarial suffix / token anomaly detection, multi-turn attack tracking, hypothetical scenario restriction, automated guardrail bypass prevention
Output Guardrails & DLP	Sensitive data (PII/API keys) redaction, canary token leak detection, hallucination & grounding verification, malicious script/URL filtering, policy violation checks, toxicity scoring
Telemetry & Governance	Real-time attack telemetry, automated session rate-limiting, SIEM & SOC log forwarding, custom heuristic rules engine, threat analytics dashboard, security audit trail

Fig 5.1 System Features

- **Prompt Injection Defense:** Focuses on detecting and neutralizing adversarial inputs that attempt to manipulate model behavior. It includes mechanisms to identify direct and indirect prompt injections, analyze semantic intent, decode obfuscated payloads (such as base64 or hex formats), filter out delimiter manipulation, strictly enforce prompt boundaries, and neutralize adversarial prefixes.

- **Jailbreak Mitigation:** Addresses methods used to bypass system restrictions or safety controls. This includes blocking roleplay and "Do Anything Now" (DAN) exploits, detecting token anomalies and adversarial suffixes, tracking multi-turn attacks across conversations, restricting hypothetical scenarios used to trick models, and preventing automated guardrail bypasses.

- **Output Guardrails & DLP (Data Loss Prevention):** Controls and filters model outputs to ensure safety and data privacy. Key features include redacting sensitive information like Personally Identifiable Information (PII) and API keys, detecting canary token leaks, verifying factual grounding to prevent hallucinations, filtering out malicious scripts and URLs and scoring text for toxicity.

- **Telemetry & Governance:** Provides real-time visibility, logging, and operational controls to maintain security compliance. It encompasses real-time attack telemetry, automated session rate-limiting, forwarding logs to SIEM/SOC platforms, maintaining a custom heuristic rules engine, monitoring threats via an analytics dashboard, and recording detailed security audit trails.

VI. EXPECTED RESULTS AND OUTCOMES

The suggested language model guardrail architecture remains in the initial blueprint and blueprint validation phase, with projected results grounded in core structural design principles and academic security research.

First, by actively scanning raw user prompts prior to model execution, the input defense layer aims to drastically lower successful direct prompt overrides and instruction hijacking. Running live contextual meaning evaluations and payload analysis ensures illegitimate commands are intercepted before entering the main model, preserving operational limits and system stability.

Second, tracking dialogue context across entire user sessions helps the platform counter progressive multi-step attack strategies, such as gradual escalations. Continuous session monitoring prevents models from being manipulated over time to ignore safety boundaries, resulting in dependable execution and persistent rule adherence.

Third, the outgoing inspection layer instantly evaluates generated responses prior to user delivery. Automatically masking confidential credentials, personal identification records, harmful URLs, and unsafe statements minimizes exposure risks and aligns with security standards.

Finally, consolidating live attack telemetry, rule violations, and audit logs produces a structured data repository for threat analysis. This ongoing intelligence feed empowers security teams to investigate emerging exploit strategies and refine defensive measures against unrecognized vulnerabilities.

EXPECTED OUTCOMES OF THE LLM JAILBREAK & PROMPT INJECTION FIREWALL

Area	Expected Outcome
Prompt Injection Defense	Detection of direct and indirect injection techniques.
Jailbreak Mitigation	Blocking of role-play and other exploit attempts.
Semantic Intent Analysis	Identification of malicious intent behind user prompts.
PII & Sensitive Data Detection	Automated redaction of personally identifiable information.
Output Guardrails	Verification of response grounding and policy checks.
Telemetry & Log Auditing	Real-time monitoring and forwarding to SIEM/SOC.
Governance & Governance	Enforcement of custom heuristic rules and rate-limiting.
Integration Resilience	Validation of system integrity and secure state detection.

VII. FUTURE SCOPE

This protective AI barrier represents a dynamic security architecture built to scale alongside shifting generative vulnerabilities, exploit methods, and underlying neural models. Looking ahead, future enhancements will likely bring self-updating live safeguards, cross-media threat detection, and instant automated mitigation for high-stakes enterprise AI systems.

- **Multi-Agent Threat Coordination:** The defense platform tracks and correlates security telemetry across several AI entities simultaneously to spot distributed attack sequences, gradual multi-turn escalations, and system-wide injection attempts, helping teams maintain unified guardrails across enterprise workflows.
- **Autonomous SOC Integration:** By linking directly with Security Operations Center (SOC) platforms, the system can automatically isolate compromised sessions, trigger dynamic rate limits, and apply emergency patches, which speeds up incident response while reducing analyst workload.
- **Adaptive Heuristic Learning:** The mechanism continuously examines novel jailbreak patterns and prompt injection strategies to automatically update safety policies, keeping core models secure against zero-day threats over long operational periods.
- Multi-Modal Defense Mechanism: Security checks extend across code, image, and audio inputs to detect hidden malicious payloads, allowing organizations to deploy multi-modal AI systems safely without risking instruction exposure or indirect hijacking.
- Model Digital Twin: By operating a mirrored replica of the primary model environment, the platform safely runs aggressive red-teaming simulations in isolation to uncover system vulnerabilities and deploy preventive patches before live rollout.

VIII. CONCLUSION

Within this study, we presented a comprehensive model guardrail architecture built to shield generative systems from security exploits throughout live enterprise execution. The proposed design merges real-time input sanitization, multi-step override prevention, and outbound data leakage controls to monitor user queries alongside generated outputs continuously. Furthermore, a core monitoring tier handles centralized logging, tailored policy enforcement, and seamless SIEM/SOC connectivity. The key benefit of this setup stems from bringing these separate security mechanisms into one cohesive system. Whereas conventional methods rely on fragmented defenses—like simple phrase blocking or static text masking—this architecture consolidates query screening, runtime protection, and central governance into a single, unified security platform.

REFERENCES

1. **Greshake, K., et al.** (2023). Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *AISec / arXiv*

2. **Zou, A., et al.** (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv*

3. **Liu, Y., et al.** (2024). Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *NDSS / arXiv*

4. **Inan, H., et al.** (2023). Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. *arXiv*

5. **Rebedea, T., et al.** (2023). NeMo Guardrails: A Toolkit for Controlling LLM Applications. *EMNLP System Demos*

6. **OWASP Foundation.** (2025). OWASP Top 10 for Large Language Model Applications (v2.0). *OWASP Project*

7. **Yi, J., et al.** (2024). Benchmarking Indirect Prompt Injections in Retrieval-Augmented Generation Systems. *arXiv*

8. **Hackett, W., et al.** (2025). Bypassing LLM Guardrails: Evasion Attacks against Detection Systems. *arXiv*

9. **Perez, F., & Ribeiro, I.** (2022). Ignore Previous Instructions: Attack Vectors via Prompt Injection in LLMs. *IEEE SPW / arXiv*

10. **Russinovich, M., et al.** (2024). Azure AI Content Safety & Prompt Shield: Protecting Large Language Models. *Microsoft Security*

