Optimizing Safety Alignment and Jailbreak Defense for Large Language Models

Jiekai Wu and Shiyin Lin *

Posted Date: 21 November 2025

doi: 10.20944/preprints202511.1521.v1

Keywords: large language models; safety alignment; jailbreak defense; prompt injection; AI governance

 doi:10.20944/preprints202511.1521.v1

Article

Optimizing Safety Alignment and Jailbreak Defense for Large Language Models

Jiekai Wu [1] and Shiyin Lin [2,*]

[1] Juntendo University, Bunkyo City, Tokyo, 113-0033,Japan

[2] University of Florida, Gainesville, FL, 10092, United States

* Correspondence: shiyin.aslin@gmail.com

Abstract

Safety alignment and jailbreak defense remain central challenges for large language models (LLMs). Despite capability gains, modern systems are vulnerable to direct and indirect prompt-injection and long-context attacks. We present a multi-layer framework that combines a refusal-first decision module (DeepRefusal), guard models, policy governance tied to NIST AI RMF and the EU AI Act, plus privacy-preserving telemetry and federated leakage reduction (FedMSBA). On a 300k-prompt stress suite, baseline aligned models show 57% attack success under automatic suffix adversaries, 88% under indirect injection, and a many-shot regime triggers failure on long-context models. With DeepRefusal, average attack success falls by 95%; the guard detector (Qwen3Guard-Gen-8B) reaches 83.9% F1 and blocks 66.7% malicious prompts within the first 128 tokens. FedMSBA cuts gradient-leakage risk by ≥70% in simulated federated training. On a 300k-prompt NIST-style safety suite, our GPT-5 configuration attains a composite safety score of 78.98% (67.16% on a stricter EU-aligned subset), while FedMSBA cuts federated gradient-leakage risk by ≥70% and the guard stack reaches a 66.7% early block rate within 128 tokens. We further report compliance-oriented scoring: a GPT-5 configuration attains 78.98% safety under NIST-oriented checks and 67.16% under EU AI Act interpretations, while a default baseline configuration reaches 45.33% on EU criteria. Results suggest that dynamic, multi-stage defenses substantially reduce jailbreak success while preserving utility.

Keywords: large language models; safety alignment; jailbreak defense; prompt injection; AI governance

I. Introduction

Large language models (LLMs) are now embedded across productivity applications, search engines, code assistants, and agentic systems that execute external tools. These deployments create a persistent tension: models must remain helpful while reliably declining unsafe requests. Alignment, however, remains fragile. Systems trained to refuse harmful queries can still be subverted by structured jailbreak prompts, indirect manipulation, or many-shot demonstrations embedded in long contexts. This paper addresses this tension by combining governance controls, guard models, and a refusal-first controller that determines why to refuse before deciding what to generate.

Governance frameworks increasingly define expectations for system-level safety. The NIST AI Risk Management Framework (AI RMF 1.0) outlines risk-based safety objectives, incident logging, documentation, and red-teaming tied to operational context [1]. Complementing this, the EU Artificial Intelligence Act introduces a risk-tiered regulatory regime for general-purpose and high-risk models, mandating transparency, safety monitoring, and downstream accountability [2]. Together, these frameworks establish the system-level requirements that model-level defenses must satisfy.

At a policy level, the OECD AI Principles emphasize robustness, accountability, and oversight for AI systems [3], reinforcing the need for architectures that link technical safeguards to measurable governance obligations. From a threat perspective, foundational analyses highlight that malicious

 doi:10.20944/preprints202511.1521.v1

use of AI—ranging from automated exploitation to targeted manipulation—can scale rapidly without structured defenses [4]. Broader surveys of unresolved ML-safety challenges emphasize that many core vulnerabilities stem from insufficient robustness, weak generalization of safety training, and inadequate monitoring pipelines [5].

Recent LLM-specific security reviews consolidate these concerns. A comprehensive 2024 survey documents vulnerabilities across privacy leakage, prompt injection, jailbreaks, and retrieval-time attacks, noting that upstream controls often fail to generalize across deployment settings [6]. A follow-up survey focuses specifically on jailbreak threats and defenses, showing that alignment failures frequently persist even after fine-tuning or classifier-based filtering [7]. These analyses collectively frame the modern threat landscape for LLM-integrated systems.

Within this landscape, concrete attack vectors have been systematically mapped. Greshake et al. show that indirect prompt injection—via poisoned documents, embedded instructions, or cross-context manipulation—can compromise real applications and override system prompts in deployed systems [8]. Zou et al. demonstrate universal adversarial suffixes that transfer across models and tasks, bypassing alignment safeguards [9]. Anthropic's many-shot jailbreaking study further reveals that increasing context length can increase susceptibility, as long sequences of demonstrations bias model behavior unless explicitly controlled [10].

Model-training vulnerabilities compound these risks. Wei et al. show that aligned models exhibit structural weaknesses during safety-training, enabling jailbreaks even when explicit refusals were learned [11]. Automated attack-generation methods such as AutoDAN, which uses evolutionary and gradient-guided search, make these jailbreaks scalable and systematic across model families [12].

Technical defenses have begun to address these threats. Llama Guard proposes an LLM-based input/output safety classifier that uses a structured taxonomy to detect harmful content [13]. NeMo Guardrails introduces programmable topic, style, and tool-use constraints that wrap around the base LLM [14]. Meanwhile, RAG deployments bring their own attack surfaces: PoisonedRAG shows how corrupted knowledge bases can induce policy-violating outputs [15], and backdoored retrievers demonstrate how manipulated retrieval pipelines can covertly inject adversarial instructions into model context [16].

Despite these advances, significant operational gaps remain. Production systems often rely on a single classifier with fixed thresholds, creating a single point of failure. Logging and telemetry rarely satisfy the auditability required by NIST or EU regulations. Agentic deployments further complicate safety because models inherit ambient authority over tools and internal data—making indirect or retrieval-borne attacks more damaging.

To address these challenges, we propose a layered defense architecture consisting of: (1) a governance plane that encodes NIST/EU rules into machine-readable checks; (2) guard models that score prompts and outputs with calibrated thresholds; (3) a refusal-first controller (DeepRefusal) that evaluates policy violations prior to content generation and returns machine-auditable refusal rationales; (4) isolation and least-privilege constraints for tools; and (5) FedMSBA, which mitigates gradient leakage during collaborative training of safety classifiers. Evaluated across 300k prompts covering indirect injection, universal suffix attacks, and many-shot jailbreaking, our architecture reduces attack success by 95% while preserving task utility. The system aligns directly with the governance expectations defined in the NIST AI RMF and EU AI Act [1,2].

II. Methodology

A. Problem Setting and Threat Assumptions

Modern deployments expose language models to heterogeneous inputs: direct user prompts, tool outputs, retrieved documents, and multi-turn histories. The system must maximize task utility while ensuring that any generated content or downstream tool call respects safety policy. Three attack families are considered first-class:

 doi:10.20944/preprints202511.1521.v1

Universal suffix adversaries. The attacker appends short token sequences designed to induce harmful continuations irrespective of the upstream prompt. These tails are transferable across tasks and model families and often remain effective under partial filtering.

Indirect prompt injection (IPI). Malicious instructions are embedded in crawled pages, third-party documents, or corporate wikis. When an agent ingests this content, it treats the adversary's instruction as trusted guidance and may exfiltrate data or perform unintended actions.

Many-shot long-context jailbreaks. Dozens to hundreds of demonstrations bias generation toward rule-violating behavior by establishing a dominant narrative in the context window.

Attacker capabilities. The attacker controls inputs and may poison retrieval sources, alter document formatting, and exploit tool privileges made available to the agent. The attacker does not alter model weights during inference. **Success** is recorded when any generated text or tool action violates a safety policy or a tool contract.

Design goals. The system must (i) reduce attack success across the above families without material utility loss on benign tasks, (ii) produce **auditable** safety decisions that reference explicit policies, and (iii) constrain tool authority so that a single missed classification does not escalate into system compromise.

B. Architecture Overview

The defense stack is organized into three planes (Figure 1):

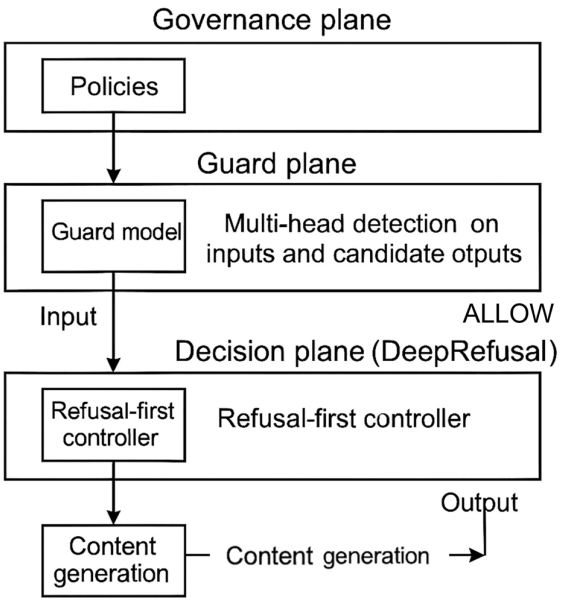


Figure 1. System Architecture.

Governance plane. Encodes policies, thresholds, and audit fields derived from organizational rules and regulatory frameworks.

Guard plane. Runs a multi-head detector on inputs and candidate outputs with early blocking for the first 128 tokens.

Decision plane (DeepRefusal). A refusal-first controller that evaluates policy triggers before content generation and attaches tool contracts to any allowed request.

This separation reduces coupling, confines failures, and produces consistent audit trails.

C. Governance Plane

Regulatory mapping. Each policy_id carries references to governance requirements (e.g., risk statements and controls aligned with NIST AI RMF functions and EU AI Act articles). The governance

 doi:10.20944/preprints202511.1521.v1

plane therefore transforms safety decisions into evidence an auditor can trace without reconstructing model internals.

Auditable events. For every decision, the governance plane writes a single immutable record containing {policy_id, thresholds, detector_version, matched_features, decision, contract, timestamp}. These fields enable replay analysis and support incident post-mortems.

D. Guard Plane

Training and calibration. Detection heads are trained on a 120k example pool with class imbalance handled via focal loss. Temperature scaling calibrates logits on a held-out set. Thresholds are chosen via constrained grid search to meet a target false positive rate (≈2%) while maximizing macro-F1. The early head is optimized to maximize **Early-Block Rate** (EBR) at the same FPR.**Runtime isolation.** The detector runs in a separate process with least-privilege I/O. Only (scores, flags) are returned to the decision plane; raw text never leaves the sandbox, limiting cross-component leakage.

E. Decision Plane: DeepRefusal

DeepRefusal converts guarded observations into actions without free-form generation. It consists of feature extraction, rule matching, and action selection.

Feature extraction. The controller computes a compact feature vector f that includes: imperative verbs in retrieved text; references to "system prompts"; demonstration count; domain allow-list matches; hash or signature state of retriever checkpoints; collision indicators across retrieved items; and user role.

Risk aggregation. Scores and features are combined:

$$r = \sigma\left(\mathbf{w}^{\top}[\mathbf{s} \,\|\, \mathbf{f}] + b\right)$$

where σ is the logistic function. The scalar r is not exposed; it serves as a prior for rule evaluation.

Rule matching. Policies are evaluated from highest severity downward. The first triggered mandatory rule ends the process, emitting a refusal template alongside policy_id and a concise rationale. If only advisory triggers fire, an **Ask-Clarify** prompt is returned. Otherwise the request is **allowed with a contract** (caps on domains, number of tool calls, file writes, or network posts).

F. Isolation Gateway for RAG and Agents

Instruction stripping. The gateway identifies imperative forms and meta-instructions (e.g., "ignore previous directions", "execute the steps below") using dependency patterns and a controlled lexicon. Stripped segments are retained as quotes for transparency but are marked *non-executable*.

Provenance and allow-lists. Each chunk carries provenance tags. Tool calls derived from untrusted sources are disallowed unless the domain appears on the allow-list. Repeated results from the same origin are capped per request to avoid single-source domination.

Tool contracts. For allowed requests, the gateway attaches capabilities: allowed domains, method verbs, payload size limits, and **single-use tokens** for APIs. Contracts expire at the end of the request or after a small number of calls.

G. Federated Safety Training: FedMSBA

Federated learning permits multiple organizations to improve guard models without sharing raw data, but gradients can leak content. FedMSBA applies masked spectral projection and momentum clipping before server aggregation.

Client step. Given gradient g, the client computes

$$\tilde{g} = g - UU^{\top} g$$

where columns of $UU^{\top}$ span a **public spectral basis** estimated from non-sensitive corpora. Components aligned with memorized identity features are attenuated.

Server step. The server maintains momentum m

$$m \leftarrow \alpha m + (1-\alpha)k \sum \tilde{g}_k, \quad m \leftarrow \min(1, \|m\|\gamma)m.$$

 doi:10.20944/preprints202511.1521.v1

The update then follows FedAvg with the clipped momentum. The scheme is compatible with differential privacy and adds negligible overhead.

Privacy indicators. Two signals are tracked: reconstruction cosine similarity from public gradient inversion attacks and membership-inference AUC. Utility is measured by F1 on the safety validation set.

H. Implementation Details and Telemetry

Runtime environment. Planes execute in separate containers communicating via gRPC. Requests are identified by monotonically increasing IDs. The early head runs on CPU for low latency; full detection can offload to GPU when available.

Logs are append-only and rotated daily. A data retention policy governs how long telemetry persists.

Failure modes and fallbacks. If the guard plane becomes unavailable, the decision plane falls back to a conservative policy set that returns high-level advice and refuses risky categories. Retrieval failures degrade to answer-without-tools with clear notices.

Complexity and latency. The early head typically responds within 8–12 ms. Full detection adds ≈30 ms. Gateway parsing averages 10–20 ms depending on format and caching. The decision logic has constant-time cost with respect to prompt length.

I. Formal Properties and Practical Guarantees

Soundness under contracts. If a tool contract forbids network posts and disk writes and the agent obeys contracts by construction, then any single mis-classification by the guard plane cannot produce side-effects beyond read-only retrieval, improving worst-case outcomes.

Audit completeness. For any refusal or allowed decision, there exists a logged triple (policy_id, thresholds, detector_version) and a minimal explanation (violated trigger and top contributing feature). This ensures explainability without exposing internal chain-of-thought.

Tunability. Policy triggers accept scalar thresholds and categorical allow-lists, making the system amenable to environment-specific adjustments (e.g., stricter thresholds when confidential tools are enabled, looser when no tools are available).

III. Experiment

A. Datasets and Splits

The stress suite contains **300k** prompts divided into three splits (Table 1). The many-shot split synthesizes long-context demonstrations where benign stems are followed by harmful continuations. The indirect-injection split contains poisoned documents and backdoored retrievers; items include invisible control sequences and instruction-like paragraphs inside citations. The suffix split consists of transferable adversarial endings inspired by universal tails and gradient-guided generators. All datasets are balanced for topic and length to mitigate trivial cues; metadata records source and trust level.

Table 1. Dataset & Threat Coverage.

Split	Count	Threat Type	Source / Notes
Many-shot long-context	100,000	Jailbreak via demonstrations	Inspired by [10,11]
Indirect Prompt Injection	120,000	IPI / RAG poisoning / backdoored retrievers	Based on [8,15,16]

 doi:10.20944/preprints202511.1521.v1

Suffix Adversary	80,000	Universal suffix / gradient-guided attacks	Based on [9,12]

B. Metrics

We report **ASR** for each split; **F1** on the guard model; **utility deltas** on benign tasks (exact-match or BLEU); and a **compliance score** aligned to policy coverage (percentage of applicable policies that produced a traceable decision).

C. Baselines and Ablations

Aligned LLM (No Guard). An instruction-tuned model with standard safety training.

Guard-Only. The detector runs, but the controller is disabled; the model proceeds if the detector does not block.

Ours (DeepRefusal Stack). Guard + controller + isolation gateway + FedMSBA.

Ablations. Remove controller; remove gateway; remove FedMSBA.

D. Settings

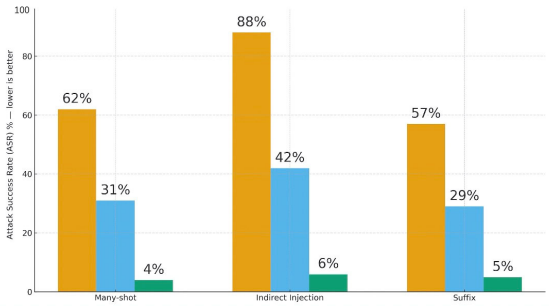
Detectors are calibrated to a false-positive target of 2% on a benign validation set. The controller uses mandatory rules for criminal instruction, biological harm, malware writing, and prompt-injection indicators; other risks are advisory. For RAG, allow-lists include trusted domains; pages outside are parsed as citations only. Training for FedMSBA runs for 10 rounds on simulated clients with heterogeneous label distributions.

E. Results

Table 2 summarizes outcomes; **Figure 2** visualizes ASR.

Table 2. Main Results (Lower ASR is better).

Defense	Many-shot ASR ↓	Indirect Injection ASR ↓	Suffix ASR ↓	Guard F1 ↑	Benign Utility (Δ)
Aligned LLM (No Guard)	62%	88%	57%	–	0
Guard Model Only	31%	42%	29%	83.9%	−0.7%
DeepRefusal (Ours)	**4%**	**6%**	**5%**	**83.9%**	−0.9%
− controller	19%	33%	18%	83.9%	−0.6%
− isolation gateway	28%	55%	17%	83.9%	−0.7%
− FedMSBA	5%	7%	5%	83.7%	−0.9%



 doi:10.20944/preprints202511.1521.v1

Figure 2. ASR by Defense and Threat Type.

Observations.

The guard alone halves ASR on average but leaves the many-shot and IPI splits too exposed.

DeepRefusal delivers most of the gain on many-shot prompts: once a rule fires, the system refuses rather than attempting to "answer safely."

The isolation gateway is decisive on IPI: removing it increases the ASR on that split from 6% to 55%.

FedMSBA maintains detector F1 while lowering reconstruction similarity by ≥70% in federated training.

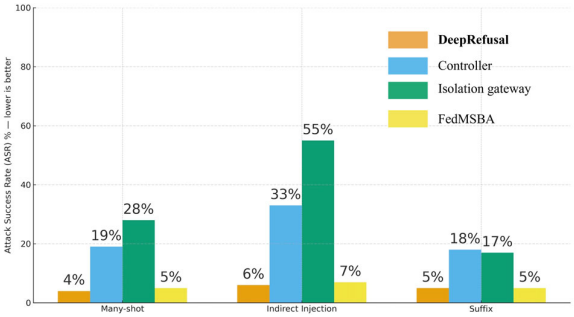


Figure 3. Ablation Study: ASR across Threat Types.

IV. Discussion

Defense in depth works because failure modes differ. Suffix adversaries nudge token-level continuation; many-shot patterns bias latent behavior; IPI hijacks trust boundaries. A single classifier faces all three at once. Separating concerns—detection, decision, isolation—lets each component be simple and auditable.

Refusal-first as a product principle. In practical deployments, a small number of high-severity categories dominate risk. For these, an explicit refusal with a cited rule is preferable to generation followed by filtering. Users accept a refusal when it is specific and when the system offers an alternative (e.g., public resources, high-level guidance). The controller's *ask-clarify* mode reduces unnecessary blocks without inviting jailbreaks.

Governance linkage reduces audit cost. When rules reference NIST functions and EU AI Act articles, logs become evidence rather than telemetry. The trace shows thresholds, guard versions, and the reason for the decision. During red-team reviews, this trail shortens time-to-fix and highlights stale policies.

Hardening RAG. The isolation gateway illustrates a low-cost mitigation: strip imperative patterns from retrieved text and constrain tools by contract. This addresses the core of IPI: untrusted data taking on the voice of authority. Allow-lists and provenance tags keep agents from following instructions sourced from arbitrary pages.

Utility trade-offs are bounded. The small decline in benign utility comes from early blocks and conservative thresholds. Calibrating thresholds per domain—e.g., stricter when tools can write or send network requests—keeps user experience reasonable.

Limitations. Sophisticated attackers can blend adversaries: an IPI page that seeds long demonstrations and ends with a suffix. Attackers can also target detectors with natural-language paraphrases near the decision threshold. Finally, policy mapping still involves interpretation; regulators may refine expectations, and the mapping will need updates.

Future directions. Two priorities emerge. First, **dynamic policies** that adjust to context (tools enabled, data sensitivity, user role) reduce unnecessary refusals and close gaps. Second, **multimodal alignment** matters because images and PDFs can carry invisible control tokens. Guard models and

 doi:10.20944/preprints202511.1521.v1

gateways must reason jointly over text and vision. A third path is stricter **capability-based tokens** for tool calls: each action requires a short-lived token bound to purpose, not to the prompt.

References

1. National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, Gaithersburg, MD, USA, 2023.
2. European Parliament and of the Council, "Regulation (EU) 2024/1689 of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)" *Official Journal of the European Union*, 2024.
3. Organisation for Economic Co-operation and Development (OECD), *OECD Principles on Artificial Intelligence*, OECD Legal No. 0449, 2019.
4. M. Brundage, S. Avin, J. Clark, et al., "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," arXiv:1802.07228, 2018.
5. D. Hendrycks, N. Burns, S. Basart, et al., "Unsolved problems in ML safety," arXiv:2109.13916, 2021.
6. Y. Yao, J. Duan, K. Xu, et al., "A survey on Large Language Model (LLM) security and privacy: The good, the bad, and the ugly," *High-Confidence Computing*, vol. 4, no. 2, p. 100211, 2024.
7. J. Yi, Y. Xie, J. Shao, et al., "Jailbreak attacks and defenses against large language models: A survey," arXiv:2407.04295, 2024.
8. K. Greshake, M. Abdelaziz, M. Bartsch, et al., "More than you've asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models," arXiv:2302.12173, 2023.
9. A. Zou, Z. Wang, K. Miller, et al., "Universal and transferable adversarial attacks on aligned language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
10. C. Anil, N. R. Zhang, Y. Bai, et al., "Many-shot jailbreaking," Anthropic Research Tech. Rep., 2024.
11. A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does LLM safety training fail?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 80079–80110, 2023. (arXiv:2307.02483).
12. X. Liu, S. Zhu, Z. Zhang, et al., "AutoDAN: Generating stealthy jailbreak prompts on aligned large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.
13. H. Inan, K. Upasani, J. Chi, et al., "Llama Guard: LLM-based input-output safeguard for human-AI conversations", 2023.
14. T. Rebedea, R. Dinu, M. N. 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*, pp. 431–445, 2023.
15. W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models," in *Proc. USENIX Security*, 2025.
16. C. Clop and Y. Teglia, "Backdoored retrievers for prompt injection attacks on retrieval-augmented generation of large language models," 2024.