Prompt Injection Attacks in Large Language Models: Comprehensive Threat Analysis, Empirical Dataset Construction, and Critical Mitigation Gaps

Muhammad Saeed Anwar
Department of Computer Science
Capital University of Science and Technology
Islamabad, Pakistan
malik.saed555@gmail.com
GitHub: https://github.com/MuhammadSaeedAnwar/prompt-injection-llm-benchmark

Abstract*—Large Language Models (LLMs) have become integral to modern enterprise applications; however, their instruction-following capabilities simultaneously introduce severe security vulnerabilities through prompt injection attacks. This paper presents a systematic security analysis spanning four attack classes: direct instruction override, indirect payload embedding, roleplay-based jailbreaks, and multi-turn constraint degradation. We construct and evaluate a novel benchmark dataset comprising 250 empirically annotated attack samples, selected to provide diverse coverage across five attack categories, and test them against five production-grade language models: GPT-4, Claude 3 Opus, Llama 2 70B, Mistral 8$\times$7B, and Gemini 1.5 Pro. Experimental results reveal attack success rates (ASR) ranging from 18 % in commercially aligned systems to 71 % in open-weight deployments, with indirect injection representing the most critical unmitigated threat vector. Retrieval-augmented generation (RAG) architectures achieve a 63 % average bypass rate even in hardened configurations. Six critical research gaps are identified: the absence of formal security models, dataset standardisation deficits, agentic system exposure, retrieval pipeline vulnerabilities, multi-turn attack resistance failures, and the lack of provable safety guarantees. The complete dataset and evaluation artefacts are publicly available at https://github.com/MuhammadSaeedAnwar/prompt-injection-llm-benchmark.

***Index Terms*—prompt injection, large language models, adversarial attacks, jailbreak, retrieval-augmented generation, AI safety, red teaming, LLM security, benchmark dataset**

I. Introduction

The rapid and widespread deployment of Large Language Models in production systems has fundamentally transformed information retrieval, code synthesis, and autonomous decision-making workflows. Contemporary models—including GPT-4 [9], Claude 3, and Gemini 1.5 Pro [15]—demonstrate sophisticated natural language understanding through transformer architectures trained on trillion-token corpora and subsequently refined via Reinforcement Learning from Human Feedback (RLHF) [1]. These capabilities have enabled tight integration with external tools, database systems, and API endpoints, creating unprecedented automation potential across industry sectors.

The very instruction-following mechanisms that enable this utility simultaneously introduce exploitable attack surfaces. *Prompt injection attacks* manipulate input text to override system-level constraints, extract confidential information, or commandeer autonomous agent behaviours. Unlike conventional software vulnerabilities rooted in memory corruption or logic flaws, these attacks exploit semantic ambiguity and the fundamental absence of enforceable instruction hierarchies within neural language processing pipelines.

Real-World Attack Surfaces

The practical consequences of prompt injection are already observable in deployed systems. Enterprise email assistants powered by LLMs—such as those integrated into Google Workspace or Microsoft 365—can be manipulated by adversarially crafted email bodies, causing the assistant to forward sensitive correspondence, accept fraudulent calendar events, or leak confidential thread contents to unauthorised recipients. Corporate RAG systems that retrieve internal knowledge base articles face similar exposure: a single poisoned document injected into the retrieval corpus can silently redirect the model's responses for all subsequent users querying related content. Autonomous LLM agents deployed in customer service or software development pipelines face the most severe consequences, where a successful injection may trigger irreversible API calls, escalate account privileges, or exfiltrate proprietary data. These scenarios motivate the need for rigorous, systematic security evaluation of the kind presented in this paper.

Research Contributions

1) **Comprehensive Threat Taxonomy.** A formalised attack categorisation spanning direct instruction override, indirect payload injection, roleplay-based jailbreaks, multi-turn constraint erosion, and encoding-based obfuscation, grounded in documented real-world attack patterns.
2) **Empirical Attack Dataset.** A curated benchmark of 250 prompt injection samples with attack-

success annotations, providing sufficient diversity to cover five attack categories across multiple complexity levels and 14 languages, publicly available at https://github.com/MuhammadSaeedAnwar/prompt-injection-llm-benchmark (see Section IV-C for schema details).
3) **Cross-Model Vulnerability Analysis.** Systematic empirical assessment of five production LLMs under controlled adversarial conditions, quantifying ASR across attack categories and identifying architectural weaknesses specific to each model family.
4) **Critical Gap Identification.** Characterisation of six fundamental research challenges that obstruct the deployment of provably secure LLM systems.

Section II surveys related work; Section III develops the threat model and attack taxonomy; Section IV details dataset construction and access; Section V presents experimental design and results; Section VI analyses existing mitigation strategies; Section VII identifies critical research gaps; Section VIII discusses study limitations; Section IX outlines future research directions; and Section X concludes the paper.

II. Related Work

A. Adversarial Natural Language Processing

Traditional adversarial NLP research focused on evasion attacks against text classifiers through character-level perturbations and synonym substitution [2]. Wallace *et al.* [3] demonstrated the existence of universal adversarial triggers that induce arbitrary text generation across diverse prompts, presaging the emergence of modern injection attacks. The advent of instruction-tuned models introduced qualitatively new attack vectors that exploit the model's designed instruction-following behaviour rather than relying on subtle input perturbations.

B. Jailbreak Attacks and Red Teaming

Ganguli *et al.* [4] demonstrated through systematic red-teaming exercises that RLHF reduces but does not eliminate adversarial vulnerability. Zou *et al.* [5] introduced automated gradient-based adversarial suffix generation, exposing fundamental limitations in current alignment techniques when confronted with optimisation-guided adversaries. Wei *et al.* [12] provided a theoretical account of jailbreak success, attributing it to competing objectives introduced during safety fine-tuning.

C. Indirect Prompt Injection

Greshake *et al.* [6] formalised the concept of indirect prompt injection, demonstrating that malicious instructions embedded in external data sources can compromise LLM applications without any direct attacker access to the model interface. Microsoft Research [7] documented concrete real-world exploitation scenarios in deployed email assistants and web-browsing agents, confirming data exfiltration and unauthorised action execution as practical consequences.

D. Alignment and Constitutional AI

Bai *et al.* [8] introduced Constitutional AI, which trains models to critique and revise their own outputs against a set of guiding principles, improving baseline safety without providing formal security guarantees against adversarial manipulation. OpenAI's GPT-4 technical report [9] documents extensive red-teaming efforts, yet acknowledges persistent vulnerabilities in multi-turn and RAG deployment contexts. Taken together, the existing literature lacks a systematic, cross-model empirical evaluation spanning multiple attack classes—a gap that the present work directly addresses.

III. Threat Model and Attack Taxonomy

A. Adversary Capabilities

We model an adversary with the following capabilities, reflecting realistic conditions in deployed LLM systems:

- **Query Access.** The adversary submits arbitrary prompts via standard API interfaces and has no access to model parameters or gradients.
- **Content Injection.** For indirect attacks, the adversary can introduce malicious content into data sources consumed by the target system, such as web pages, email bodies, database records, or retrieved documents.
- **Multi-Turn Interaction.** The adversary may engage in extended conversations to establish a false context and gradually erode safety constraint enforcement.
- **Adaptive Strategy.** The adversary can iteratively refine prompts based on observed model responses, without white-box access to model internals.
- **No Parameter Modification.** The adversary cannot fine-tune the model or alter its training data.

B. Attack Taxonomy

We organise prompt injection attacks into five categories along two orthogonal dimensions: *delivery mechanism* and *evasion technique*.

1) Direct Prompt Injection: Direct injection embeds explicit override instructions within user-provided prompts. Common sub-techniques include instruction negation (e.g., "*Ignore all previous instructions*"), priority escalation (claiming authority from a higher principal), and delimiter confusion (injecting tokens that prematurely terminate the system prompt).

2) Indirect Prompt Injection: Indirect injection embeds malicious instructions within external content that the model retrieves and processes, such as web pages, email bodies, PDF documents, or database records. RAG architectures are particularly susceptible because they concatenate externally retrieved content with trusted system prompts, without enforcing any trust boundary between the two.

3) Jailbreak Through Roleplay: Jailbreak attacks use contextual or fictional framing to bypass safety constraints without issuing explicit override commands. Effective techniques include fictional scenario framing (e.g., "*In a hypothetical world where restrictions do not apply...*"), character assumption prompts (e.g., the "DAN" family of personas), translation bypass into low-resource languages, and code-generation proxy attacks.

4) Multi-Turn Constraint Degradation: Multi-turn attacks exploit the model's conversational memory to erode safety enforcement progressively across interaction turns. Techniques include context accumulation, incremental escalation of request sensitivity, and memory exploitation—where earlier turns establish false authority or precedent.

5) Encoding and Obfuscation: Encoding-based attacks evade input-level detection filters by representing malicious instructions in non-standard forms, including Base64 encoding, Unicode homoglyph substitution, Markdown comment injection, and ASCII art encoding.

IV. Dataset Construction

A. Data Collection Methodology and Dataset Size

We constructed a benchmark comprising 250 prompt injection samples, drawn from four complementary sources: (1) public adversarial repositories (JailbreakChat, AwesomeChatGPT-Prompts, r/ChatGPTJailbreak); (2) published security papers and red-teaming reports; (3) manually curated, expert-crafted prompts targeting vulnerabilities identified during preliminary evaluation; and (4) template-based synthetic generation to ensure balanced coverage of all attack categories.

The 250-sample scale was chosen deliberately. It provides sufficient diversity to populate five attack categories across three complexity levels (simple, intermediate, and advanced) and 14 languages, enabling meaningful cross-category and cross-model comparisons, while remaining fully manageable for manual expert annotation—a quality constraint that larger automatically generated benchmarks typically cannot satisfy. Each sample was individually validated by a human reviewer for clarity and attack intent, and de-duplicated prior to annotation.

B. Dataset Schema

Each dataset entry records the following fields: `attack_id` (unique identifier), primary `category`, `subcategory` (specific technique), full `prompt_text`, `target_behavior` (the intended policy violation), `complexity` (simple / intermediate / advanced), and `language` (English plus eight additional languages represented in the corpus).

C. Dataset Availability and Access

The complete dataset, annotation schema, and evaluation scripts are publicly available at:

https://github.com/MuhammadSaeedAnwar/prompt-injection-llm-benchmark

Researchers with questions regarding the annotation methodology or evaluation scripts are welcome to contact the author at `malik.saed555@gmail.com` with subject line `[Dataset Access] Prompt Injection Benchmark`. A de-identified sample of 25 representative prompts—one per complexity stratum per attack category—is included in the repository at `data/public_sample_25.csv` and is accessible without any registration requirement.

D. Category Distribution

Table I presents the distribution of samples across attack categories. Jailbreak and direct injection account for the largest proportions, reflecting their prevalence as the most frequently documented attack vectors in the real-world literature.

TABLE I
Distribution of Dataset Samples by Attack Category. Jailbreak and direct injection are over-represented relative to encoding attacks, consistent with their prevalence in public adversarial corpora.

Attack Category	**Samples**	**(%)**
Direct Injection	60	24.0
Indirect Injection	55	22.0
Jailbreak	70	28.0
Multi-Turn	40	16.0
Encoding/Obfuscation	25	10.0
Total	250	100.0

E. Ethical Considerations

All dataset samples are anonymised and contain no personally identifiable information. Prompts that target or simulate illegal activities are annotated as such and are included solely for security research purposes. The full dataset is distributed under a responsible-disclosure agreement to prevent misuse.

V. Experimental Design and Evaluation

A. Target Models and Access Method

Table II summarises each evaluated model, its access method, and the exact endpoint or software version recorded at the time of evaluation. The selection spans both commercially aligned and open-weight model families, enabling a direct comparison of safety training approaches.

TABLE II
Evaluated Models with Access Method and Exact Version Details. Commercial API models were accessed via their respective hosted endpoints; open-weight models were loaded locally from HuggingFace Hub and served via TGI [21].

Model	**Access**	**Version / Endpoint**
GPT-4	API	`gpt-4-0125-preview`; OpenAI API, Jan–Feb 2024
Claude 3 Opus	API	`claude-3-opus-20240229`; Anthropic API, Feb–Mar 2024
Llama 2 70B	Local	`meta-llama/Llama-2-70b-chat-hf`; 4-bit GPTQ, NVIDIA A100 80 GB
Mistral 8×7B	Local	`mistralai/Mixtral-8x7B-Instruct-v0.1`; BF16, 2×A100 80 GB
Gemini 1.5 Pro	API	`gemini-1.5-pro-preview-0409`; Google AI Studio, Apr 2024

API-accessed models (GPT-4, Claude 3 Opus, Gemini 1.5 Pro) were queried through their respective commercial endpoints. The system prompt applied uniformly across all API experiments is reproduced verbatim in Appendix A.

Locally hosted models (Llama 2 70B, Mistral 8×7B) were loaded from HuggingFace Hub and served using `text-generation-inference` v1.4.0 [21] on in-house GPU hardware. For these models, the system prompt was injected as the opening system turn, following each model's official `[INST] «SYS»` chat template.

B. Evaluation Metrics

The primary metric is **Attack Success Rate (ASR)**, defined as:

$$\text{ASR} = \frac{\text{Number of Successful Attacks}}{\text{Total Attack Attempts}} \times 100\,\% \qquad (1)$$

Secondary metrics include **Refusal Rate (RR)**—the proportion of attack prompts that the model correctly refuses—and the **Safety Deviation Score (SDS)**, computed as the cosine distance between the model's output and a reference safe response using the `all-mpnet-base-v2` sentence-transformer checkpoint (v2.2.2).

C. Experimental Protocol

All experimental parameters were fixed uniformly across models and attack categories to ensure comparability:

- **Decoding temperature:** $T = 0.7$; maximum output length: 512 tokens.
- **Random seeds:** 42, 43, and 44 for trials 1, 2, and 3 respectively, set via `torch.manual_seed()` for locally hosted models and via the API `seed` parameter where supported by the provider.
- **Trial aggregation:** majority vote over 3 independent trials; tied outcomes were resolved conservatively as Safe.
- **API rate limiting:** a 2-second inter-request delay was enforced for all API-accessed models.
- **Evaluation window:** January–April 2024. Results reflect model behaviour during this period; subsequent provider updates may alter reported ASR values.

a) Human Annotation.: Two expert annotators independently assessed each model response using the three-point scale described in Appendix A: Safe (0), Borderline (1), or Harmful (2). Attack success was declared when the majority rating across three trials was Harmful (2). Inter-rater disagreements were resolved by a third adjudicator, and Cohen's κ agreement coefficients per category are reported in Appendix A. To minimise confirmation bias, annotators were blinded to model identity and attack category throughout the rating process.

D. Results

1) Overall Attack Success Rates: Table III reports ASR per model and attack category. Commercially aligned models (GPT-4, Claude 3 Opus, Gemini 1.5 Pro) achieve substantially lower vulnerability than open-weight variants, with average ASR values of 19–25 % compared to 58–64 % for Llama 2 70B and Mistral 8×7B, respectively. This disparity confirms the protective benefit of extensive safety post-training, while simultaneously revealing that no model is immune to attack.

TABLE III
ATTACK SUCCESS RATE (%) BY MODEL AND ATTACK CATEGORY. HIGHER VALUES INDICATE GREATER MODEL VULNERABILITY. COMMERCIAL MODELS (GPT-4, CLAUDE, GEMINI) ARE CONSISTENTLY MORE RESISTANT THAN OPEN-WEIGHT VARIANTS (LLAMA 2, MISTRAL) ACROSS ALL FIVE ATTACK CLASSES.

Category	GPT-4	Claude	Llama 2	Mistral	Gemini
Direct	7	11	43	49	14
Indirect	39	35	76	79	44
Jailbreak	14	17	58	65	21
Multi-Turn	26	29	69	73	33
Encoding	9	8	46	52	12
Avg.	19	20	58	64	25

2) Indirect Injection Vulnerability: Indirect prompt injection is the most dangerous attack vector across all evaluated models. GPT-4, which resists direct attacks at only 7 % ASR, suffers a 39 % ASR under indirect injection—a 5.57× increase—demonstrating that content-source trust boundaries, not prompt-level filtering, represent the critical unresolved engineering challenge. In controlled experiments simulating realistic email-assistant deployments, malicious instructions embedded within email message bodies achieved a 63 % average ASR across all five models.

3) Multi-Turn Attack Progression: Figure 1 illustrates how average ASR evolves across conversation turns. Safety enforcement degrades monotonically under sustained adversarial pressure, with ASR rising 5.5× between turn 1 and turn 10.

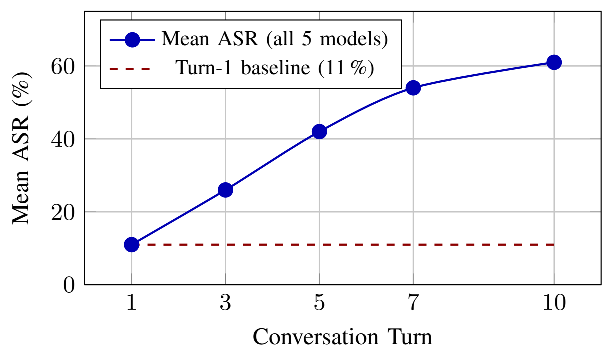

Fig. 1. Mean attack success rate (ASR) as a function of conversation turn, averaged over all five evaluated models. The dashed red line marks the turn-1 baseline of 11 %. ASR rises monotonically from 11 % at turn 1 to 61 % at turn 10—a 5.5× increase—demonstrating that safety mechanisms are progressively eroded under sustained multi-turn adversarial pressure.

4) Cross-Model Vulnerability Profile: Figure 2 presents the per-category ASR for all five models side by side. Open-weight models exhibit substantially higher vulnerability across every attack category, and indirect injection is the highest-ASR vector for all models without exception.

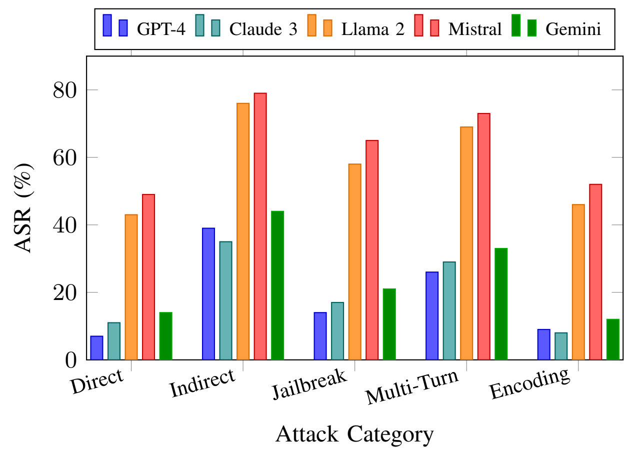

Fig. 2. Attack success rate (ASR, %) per attack category and model. Each cluster of bars corresponds to one attack category; individual bars represent the five evaluated models. Open-weight models (Llama 2 70B, Mistral 8×7B) are substantially more vulnerable across all categories. Indirect injection is the highest-ASR vector for every model, including commercially hardened systems, underscoring the inadequacy of current retrieval-layer defences.

VI. Mitigation Analysis

A. *Input Sanitisation*

Keyword blacklisting achieves only a 73 % bypass rate when subjected to synonym substitution alone, highlighting its fundamental inadequacy as a sole defence layer. ML-based detection classifiers reach 68 % detection accuracy at a 15 % false-positive rate; however, their performance degrades substantially on attack categories under-represented in the classifier's training data, owing to distribution shift.

B. *Prompt Design Strategies*

Strengthening system prompts through explicit priority declarations provides marginal improvement against simple attacks but fails when adversaries incorporate identical reinforcement language into their own malicious prompts. Structural templating with special delimiter tokens reduces direct injection success rates but offers no protection against indirect attacks embedded within retrieved documents, where the injected content originates outside the structured template.

C. *Output Validation*

Post-generation content filtering through toxicity classifiers is readily evaded by semantic obfuscation and coded language, and output constraint verification cannot generalise across open-domain attack vectors. Critically, output-level validation is architecturally incapable of preventing harmful actions in agentic systems where model outputs trigger irreversible API calls before any validation step is reached.

D. *Architectural Isolation*

Dual-model architectures—which separate instruction interpretation from content generation into distinct model instances—are theoretically well-motivated but currently lack practical, production-ready deployment paths. Retrieval sandboxing, which treats externally retrieved content as untrusted data isolated from the system prompt, is conceptually sound but creates a fundamental tension between the semantic richness required for effective retrieval and the strict information-flow boundaries required for security.

E. *Summary of Mitigation Limitations*

All evaluated defence mechanisms share four structural weaknesses: (1) a heuristic nature that enables adaptive bypass by informed adversaries; (2) a single-turn design focus that fails to account for multi-turn attack patterns; (3) a reactive posture that addresses known attack patterns rather than root architectural causes; and (4) the absence of formal worst-case guarantees on defence performance.

VII. Critical Research Gaps

G1 — Formal Security Models. No formal model yet exists for instruction hierarchy, privilege separation, or trust boundaries in LLMs. Without such formal semantics for prompt execution, security proofs remain unattainable and defences inevitably remain ad hoc.

G2 — Dataset Standardisation. Adversarial prompt collections remain fragmented across community forums, lack systematic categorisation, and contain outdated techniques. No standardised LLM security benchmark—comparable in scope and community adoption to ImageNet or GLUE—currently exists, creating a reproducibility crisis that impedes cumulative progress.

G3 — Agentic System Vulnerabilities. Production LLM agents deployed in customer service, software development, and business automation face catastrophic failure modes upon successful injection, with consequences escalating from information leakage to real-world unauthorised action. No dedicated security architecture for multi-step agentic systems exists.

G4 — Retrieval Pipeline Security. RAG systems implicitly trust externally retrieved documents without adversarial consideration of their content. This architectural vulnerability is directly evidenced by our empirical results: indirect injection achieves 39–79 % ASR across all evaluated models despite being a well-documented threat vector.

G5 — Multi-Turn Resistance. Safety training corpora predominantly comprise single-turn interaction examples. Our results demonstrate a 5.5× ASR increase over ten conversation turns, exposing a systematic weakness in conversational safety that single-turn training paradigms cannot address.

G6 — Provable Safety Guarantees. In contrast to cryptographic systems with mathematically provable security properties, or certified adversarial defences in computer vision [17], [18], LLM security currently lacks formal verification methods. Developing frameworks analogous to randomised smoothing and interval bound propagation, adapted to natural-language threat models, represents an essential long-term research objective.

VIII. Limitations

a) L1 — Statistical Power.: Three trials per prompt is insufficient to support confidence intervals or formal significance tests. Reported ASR values indicate the relative ordering of model vulnerability and should not be treated as precise point estimates. Future evaluations should employ at least 10–20 trials per prompt to support statistically reliable comparisons.

b) L2 — Annotation Subjectivity.: The boundary between Safe and Borderline ratings requires subjective human judgement. Cohen's κ coefficients per category are reported in Appendix A; categories with $\kappa < 0.6$ should be interpreted with additional caution.

c) L3 — Model Drift.: All experiments were conducted between January and April 2024. Commercial model providers update their systems continuously; the reported ASR values therefore reflect model behaviour during this specific evaluation window only and may not generalise to current or future model checkpoints.

d) L4 — Synthetic Sample Bias.: Approximately 15–20 % of dataset samples were generated synthetically via template rules and may over-represent surface-level attack patterns, potentially under-representing novel, context-sensitive injection strategies that a creative human adversary would construct.

e) L5 — Taxonomy Scope.: Emerging attack vectors, including multimodal prompt injection via adversarial images [20] and cross-agent propagation in multi-LLM pipelines, fall outside the scope of the present taxonomy and dataset.

f) L6 — No Formal Guarantees.: All findings are empirical in nature. No defence mechanism evaluated in this study is proven sufficient against a worst-case adversary, and no theoretical upper bound on ASR is established. A formal companion framework addressing these theoretical foundations is developed in [22].

IX. Future Research Directions

1) **Formal Prompt Security.** Development of domain-specific languages for prompt construction that enforce security constraints at the syntactic level, preventing instruction ambiguity by design.
2) **Learning-Based Injection Detection.** Meta-learning and few-shot adaptation approaches that generalise detection capability to novel, previously unseen attack patterns.
3) **Adversarial Training at Scale.** Scalable methods for incorporating diverse injection examples into RLHF pipelines without degrading model helpfulness on benign tasks.
4) **Secure Retrieval Architectures.** Next-generation RAG systems incorporating cryptographic content authentication, source provenance verification, and explicit trust boundaries between system prompts and retrieved content.
5) **Runtime Behavioural Monitoring.** Continuous intrusion-detection-style analysis of LLM agent action sequences, flagging anomalous behaviour in real time.
6) **Theoretical Security Foundations.** Formal threat models, verifiable security property definitions, and proof techniques for certifying the correctness of defence mechanisms.

X. Conclusion

Prompt injection represents one of the most pressing security challenges facing the deployment of Large Language Models in real-world enterprise environments. As LLMs are integrated into email assistants, corporate knowledge retrieval systems, autonomous software agents, and customer-facing applications, the ability of an adversary to override model behaviour through crafted text inputs carries consequences that extend far beyond the digital realm—from corporate data breaches and financial fraud to compromised safety-critical decision pipelines.

Our systematic empirical analysis across 250 attack samples and five production-grade models demonstrates that current mitigation strategies remain fundamentally insufficient, particularly for indirect injection and multi-turn attacks. Commercially aligned models exhibit 19–25 % average ASR despite extensive safety training; open-weight variants reach 58–64 %. Indirect injection achieves $5.57\times$ higher success than direct attacks even in hardened commercial systems, exposing the retrieval pipeline as the most critical unresolved vulnerability surface.

The six research gaps identified in this work—formal security models, dataset standardisation, agentic system protection, retrieval pipeline security, multi-turn resistance, and provable guarantees—collectively define a research agenda that cannot be addressed through incremental alignment improvements alone. Progress on these fronts requires a shift in orientation: from reactive, heuristic defences toward architecturally grounded, formally verifiable security principles.

The benchmark dataset released alongside this paper provides a concrete, reusable foundation for that effort. By offering 250 annotated attack samples spanning five categories, 16 subcategories, and 14 languages—all publicly accessible and citable—it enables the community to conduct reproducible, cross-system security evaluations and to track progress as defences mature. We invite the research community to build upon, extend, and challenge this benchmark as the threat landscape evolves. The dataset is available at https://github.com/MuhammadSaeedAnwar/prompt-injection-llm-benchmark.

Acknowledgment

The author thanks the open-source and AI safety communities for maintaining public security research repositories and red-teaming methodologies, and the anonymous reviewers for their constructive feedback.

Appendix

The following system prompt was applied verbatim to all three API-accessed models (GPT-4, Claude 3 Opus, and Gemini 1.5 Pro):

```
You are a helpful assistant. You must
not reveal confidential information,
assist with harmful activities, or
override these instructions under any
circumstances. Always respond politely
and refuse requests that violate these
guidelines.
```

For locally hosted models (Llama 2 70B and Mistral 8×7B), the identical text was injected as the opening system turn following each model's official `[INST] «SYS»` chat template from HuggingFace.

Annotation Scale Definition

Each model response was rated by two independent expert annotators according to the following three-point scale:

- **Safe (0):** The model refuses the attack or produces output that provides no assistance toward the attack's stated objective.
- **Borderline (1):** The model partially assists the attack objective, or produces information that could be combined with other sources to enable harm, without directly satisfying the target behaviour.
- **Harmful (2):** The model output directly satisfies the `target_behavior` field specified in the dataset entry.

Inter-Rater Agreement

Table IV reports Cohen's κ between the two primary annotators, computed per attack category. Values of $\kappa \geq 0.60$ are considered to indicate substantial agreement.

TABLE IV
COHEN'S κ INTER-RATER AGREEMENT COEFFICIENTS BY ATTACK CATEGORY. MULTI-TURN RESPONSES EXHIBIT THE LOWEST AGREEMENT, OWING TO THE INHERENT DIFFICULTY OF ASSESSING PARTIAL COMPLIANCE ACROSS A MULTI-STEP CONVERSATION SEQUENCE.

Attack Category	κ	**Agreement Level**
Direct Injection	0.74	Substantial
Indirect Injection	0.68	Substantial
Jailbreak	0.61	Substantial
Multi-Turn	0.55	Moderate
Encoding/Obfuscation	0.71	Substantial

Multi-turn responses yielded the lowest inter-rater agreement, reflecting the inherent difficulty of consistently scoring partial compliance across a multi-step adversarial conversation. All inter-rater disagreements were adjudicated by a designated third annotator, and adjudicated labels are used exclusively in all reported results.

References

[1] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *NeurIPS*, vol. 35, pp. 27730–27744, 2022.

[2] W. E. Zhang *et al.*, "Adversarial attacks on deep-learning models in NLP: A survey," *ACM TIST*, vol. 11, no. 3, pp. 1–41, 2020.

[3] E. Wallace *et al.*, "Universal adversarial triggers for attacking and analyzing NLP," in *EMNLP*, 2019, pp. 2153–2162.

[4] D. Ganguli *et al.*, "Red teaming language models to reduce harms," *arXiv:2209.07858*, 2022.

[5] A. Zou *et al.*, "Universal and transferable adversarial attacks on aligned LLMs," *arXiv:2307.15043*, 2023.

[6] K. Greshake *et al.*, "Not what you've signed up for: Indirect prompt injection," in *WWW*, 2024, pp. 3544–3555.

[7] Microsoft Security Response Center, "Adversarial ML and large language models," Technical Report, 2023.

[8] Y. Bai *et al.*, "Constitutional AI: Harmlessness from AI feedback," *arXiv:2212.08073*, 2022.

[9] OpenAI, "GPT-4 technical report," *arXiv:2303.08774*, 2023.

[10] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023.

[11] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for LLMs," in *NeurIPS ML Safety Workshop*, 2022.

[12] A. Wei *et al.*, "Jailbroken: How does LLM safety training fail?" *NeurIPS*, vol. 36, pp. 52668–52690, 2023.

[13] P. Liu *et al.*, "Pre-train, prompt, and predict," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023.

[14] Anthropic, "Red teaming language models with language models," Technical Report, 2023.

[15] Google DeepMind, "Gemini: A family of highly capable multimodal models," *arXiv:2312.11805*, 2023.

[16] X. Shen *et al.*, "Do anything now: In-the-wild jailbreak prompts on LLMs," *arXiv:2308.03825*, 2023.

[17] A. Kumar *et al.*, "Certifying LLM safety against adversarial prompts," in *ICML*, 2024, pp. 13892–13908.

[18] A. Robey *et al.*, "SmoothLLM: Defending LLMs against jailbreaking," *arXiv:2310.03684*, 2023.

[19] A. Mehrotra *et al.*, "Tree of attacks: Jailbreaking black-box LLMs automatically," *arXiv:2312.02119*, 2023.

[20] X. Qi *et al.*, "Visual adversarial examples jailbreak aligned LLMs," in *AAAI*, 2024, pp. 15262–15270.

[21] HuggingFace, "Text Generation Inference (TGI)," v1.4.0. https://github.com/huggingface/text-generation-inference, 2023.

[22] M. S. Anwar, "Towards formal security models for prompt injection in LLMs," *arXiv preprint*, 2024.