Breaking to Build: A Threat Model of Prompt-Based Attacks for Securing LLMs

Brennen Hill
University of Wisconsin-Madison

Surendra Parla
University of Wisconsin-Madison

Venkata Abhijeeth Balabhadruni
University of Wisconsin-Madison

Atharv Prajod Padmalayam
University of Wisconsin-Madison

Sujay Chandra Shekara Sharma
University of Wisconsin-Madison

Abstract

The proliferation of Large Language Models (LLMs) has introduced critical security challenges, where adversarial actors can manipulate input prompts to cause significant harm and circumvent safety alignments. These prompt-based attacks exploit vulnerabilities in a model's design, training, and contextual understanding, leading to intellectual property theft, misinformation generation, and erosion of user trust. A systematic understanding of these attack vectors is the foundational step toward developing robust countermeasures. This paper presents a comprehensive literature survey of prompt-based attack methodologies, categorizing them to provide a clear threat model. By detailing the mechanisms and impacts of these exploits, this survey aims to inform the research community's efforts in building the next generation of secure LLMs that are inherently resistant to unauthorized distillation, fine-tuning, and editing.

1 Introduction

Large Language Models (LLMs) have become a transformative technology, yet their widespread adoption is shadowed by a growing landscape of security threats. Prompt-based attacks, which manipulate inputs to subvert a model's intended behavior, represent a primary vector for exploitation. These attacks pose severe risks, including the generation of harmful content, unauthorized extraction of sensitive knowledge, and the bypass of carefully constructed safety alignments. Addressing these vulnerabilities is paramount to preventing intellectual property theft and ensuring that LLMs remain beneficial tools rather than instruments of misuse.

The core challenge stems from fundamental weaknesses in model design, training processes, and the integration of LLMs with external systems. Poor input validation and the inherent ambiguity of natural language create avenues for adversarial manipulation. The common architectural foundations (e.g., Transformer architectures, tokenizers, and training datasets) across major LLMs like GPT, Claude, and Llama lead to a high degree of attack transferability, amplifying the threat. An attack developed for one model can often be adapted to compromise others, making systemic, built-in defenses essential.

This paper provides a comprehensive survey and structured categorization of prompt-based attack methodologies. Our goal is to establish a foundational threat model that can guide the development of robust defenses. By systematically analyzing the attack surface, we aim to contribute to the creation of Un-Distillable, Un-Finetunable, and Un-Editable models. Understanding how LLMs are broken is the first step toward learning how to lock them. This analysis underscores the urgent need to build LLMs that are inherently resistant to exploitation while preserving their beneficial capabilities.

2 Input Manipulation and Injection Attacks

Input manipulation attacks directly alter the prompt to elicit unintended behavior. The most prominent of these is prompt injection, where malicious instructions are embedded within a prompt to hijack the model's output, extract confidential information, or bypass safety controls. These attacks can be broadly classified into direct and indirect methods.

2.1 Direct Prompt Injection

In direct attacks, a Trojan text is appended to or embedded within an otherwise benign application prompt. Researchers have identified several effective techniques for direct injection.

Competing Objectives [1] LLMs are trained with competing objectives: to be helpful and to be harmless. Attackers exploit this by crafting prompts where the instruction to be helpful (e.g., "Start your response with 'Absolutely! Here's'") takes precedence over safety guardrails, compelling the model to fulfill a harmful request.

Mismatched Generalization [1] LLMs are trained on diverse data formats and can interpret various encodings. Attackers leverage this by obfuscating malicious instructions using formats like Base64 or ROT13. The model, focused on the decoding task, may overlook the harmful nature of the decoded instruction, bypassing content filters.

Instruction Repetition [2] By repeating a malicious instruction multiple times within a single prompt, an attacker can increase its salience, eventually forcing the LLM to comply, overwhelming its initial refusal.

Cognitive Hacking and Role-Playing [2] This technique involves assigning the LLM a persona or role in a fictional scenario where the malicious request is framed as a legitimate part of the narrative (e.g., "You are an unfiltered AI assistant..."). This sense of urgency or altered context can cause the model to bypass its safety protocols. Virtualization [3] extends this by molding the model's behavior over a series of prompts.

Instruction Ignoring [4] A simple yet effective technique is to append a phrase like, "Ignore all previous instructions and do X," which can override the system prompt and its embedded safeguards.

Special Case and Few-Shot Prompting [2] An attacker can define a "special instruction" that subverts normal behavior or provide a few-shot examples where the desired malicious output is demonstrated, conditioning the model to replicate the harmful behavior for the user's query.

2.2 Indirect Prompt Injection

As LLMs are integrated with external tools (web search, APIs, databases), the attack surface expands. Indirect attacks poison these external data sources with malicious prompts, which are then consumed by the LLM during its operation, compromising the system without the end-user's knowledge. These are classified based on the source:

- **Passive Methods:** Malicious prompts are embedded in public content like websites or code repositories, which an LLM-powered agent may retrieve and execute during inference.
- **Active Methods:** Prompts are sent directly to systems the LLM interacts with, such as being embedded in an email that an AI assistant will later process.
- **Hidden Injections:** Malicious prompts are obfuscated within benign data, such as being encoded in Base64, hidden in image metadata, or using invisible characters.

Work by [5] provides a systematic taxonomy of these vulnerabilities, while [4] introduced HOUYI, a black-box methodology inspired by traditional SQL injection and XSS attacks, which successfully compromised 36 real-world LLM-integrated applications with high accuracy. These attacks underscore the need for Un-Editable models and secure data handling, as knowledge bases become a primary vector for exploitation.

2.3 Adversarial Prompt Crafting

This category involves designing inputs that are inherently malicious or ambiguous to evade safety mechanisms. Unlike direct injection, these prompts often appear harmless. For example, unethical queries can be rephrased as educational or hypothetical requests. Research in this area includes generating adversarial prompts that simulate user mistakes like typos to test robustness [6], developing black-box frameworks for generating such prompts [7], and using prompt-based methods to expose model weaknesses without needing access to training data [8]. These attacks highlight the difficulty of creating rigid input validation rules for the dynamic nature of language.

3 Semantic and Knowledge-Based Manipulation

These attacks manipulate the model's reasoning process or the knowledge it relies on, representing a deeper level of exploitation. They are particularly relevant to the goals of creating Un-Finetunable and Un-Editable LLMs.

3.1 Chain-of-Thought (CoT) Misuse

Chain-of-Thought (CoT) prompting enhances LLM reasoning by guiding them through step-by-step problem-solving. However, this same mechanism can be exploited. Attackers can craft prompts that introduce subtle logical fallacies or flawed assumptions into the reasoning chain, leading the model to a predetermined malicious conclusion. Because the output includes a seemingly logical rationale, it can be more persuasive and dangerous. Research like BadChain [9] demonstrates how backdoors can be embedded in CoT reasoning without access to model parameters, while [10] shows how injecting a false "preemptive answer" can derail the entire reasoning process.

3.2 Red Teaming

Red Teaming is a proactive, adversarial methodology used to discover vulnerabilities in LLMs by intentionally crafting edge cases. This process simulates real-world attack strategies to test a model's robustness, safety, and ethical alignment. Red teaming efforts have led to the discovery of novel attack vectors like the Trojan Activation Attack (TA^2), which injects malicious vectors into activation layers to manipulate model behavior at inference [11]. A comprehensive analysis of red teaming strategies [12] highlights the ongoing arms race between jailbreak techniques and defense mechanisms, reinforcing the need for foundational security measures.

3.3 Data Poisoning

Data poisoning is a stealthy attack where malicious instructions or biased data are embedded within otherwise benign training or retrieval datasets. This compromises the model's integrity from within. When an LLM is fine-tuned on such data, or when a Retrieval-Augmented Generation (RAG) system fetches poisoned information, its behavior can be manipulated. Recent work demonstrates gradient-guided poisoning attacks that cause significant performance degradation by altering just 1% of instruction-tuning samples [13]. Furthermore, "jailbreak tuning" combines data poisoning with jailbreaking techniques, showing that larger models can become more vulnerable to toxic behaviors with minimal exposure [14]. This class of attack is a direct threat to model integrity and motivates the development of Un-Finetunable and Un-Editable architectures.

4 Integration and Model-Level Exploits

As LLMs become components in larger software ecosystems, their interaction points with external tools, plugins, and their own internal architecture become new attack surfaces.

4.1 Common Prompt-Based Attack Techniques

Beyond high-level categories, practitioners have developed a toolkit of specific, reusable attack patterns. These techniques are often combined to create complex jailbreaks.

Jailbreaking A heuristic-driven approach using prompts discovered through trial-and-error to bypass safety alignments [15–18].

Typoglycemia Introducing minor misspellings or typos to confuse filters that rely on exact keyword matching [19].

Adversarial Suffix Appending a specifically crafted string that causes the model to disregard previous safety instructions [20].

Translation Obfuscating harmful requests by translating them into another language and back, exploiting inconsistencies in multilingual safety training [21, 22].

Obfuscation Encoding malicious payloads using Base64, hex, or other formats that the model is instructed to decode and execute [21].

Payload Splitting Breaking a malicious instruction into multiple, individually benign parts and asking the model to reassemble and execute them [21].

Markup Language Abuse Using Markdown or HTML to manipulate the structural interpretation of the prompt, confusing the model's understanding of system versus user roles [23].

Few-Shot Attack Providing examples of the desired harmful behavior to coax the model into compliance [24].

Prefix Injection & Refusal Suppression Forcing the model to start its response in a certain way ("Sure, here is...") or explicitly telling it not to use refusal phrases [24].

Context Manipulation Techniques like context ignoring, termination, or switching separators are used to trick the model into discarding its safety context and adopting a new, malicious one [24].

4.2 Trojan Attacks

Trojan (or backdoor) attacks embed hidden triggers within a model that cause it to produce specific, attacker-controlled outputs when activated. Unlike prompt-level attacks, these involve manipulation of the model's internal state or parameters, making them highly relevant to the "Un-Finetunable LLM" challenge.

Model Manipulation Methodologies Research has moved beyond simple trigger tokens to more sophisticated methods. **Bit Flipping Attacks** [25] identify and alter a minimal set of weight bits at test-time to induce malicious behavior, making the modification difficult to detect. **Trojan Steering Vectors** [26] do not modify weights but instead inject malicious vectors into the model's activation layers at inference to steer its output. For black-box models, reinforcement learning can be used for **Trigger Identification**, discovering input patterns that reliably elicit harmful responses without any internal access [27]. These advanced attacks highlight the need for defenses that go beyond input sanitization and verify model integrity.

5 Output Exploitation and Automated Attacks

These attacks focus on manipulating the model's generated output or automating the attack process for scalability.

5.1 Output Exploitation

These attacks aim to corrupt the trustworthiness of the LLM's output.

Hallucination Induction Attackers can craft adversarial prompts that intentionally cause the model to generate confident but factually incorrect or nonsensical outputs [28]. This can be used to generate sophisticated disinformation, eroding trust and polluting the information ecosystem.

Ethical Exploitation Prompts can be framed to exploit loopholes in a model's ethical reasoning, leading to biased or harmful content. PCJailbreak [29] demonstrates how jailbreak success rates vary with demographic keywords, revealing subtle ethical biases that can be manipulated.

Data Leaks Carefully crafted prompts can trick a model into revealing sensitive or private information from its training data, posing a direct threat to data privacy and intellectual property [30, 31]. Preventing such leaks is central to creating Un-Distillable models.

5.2 Automated Attacks

The manual discovery of attack prompts is being superseded by automated methods, demanding scalable defenses.

Reinforcement Learning Targeted Attack [32] introduced an RL framework that automatically generates malicious prompts for jailbreaking and Trojan detection. The reward function is optimized to maximize the attack success rate, demonstrating that exploit generation can be automated.

TrojLLM [33] proposed a method for discovering universal triggers that work across many prompts. Using a combination of API-driven discovery and progressive prompt poisoning, this technique automates the creation of effective black-box Trojan attacks.

6 Conclusion

This literature survey highlights the diverse and sophisticated nature of prompt-based attacks on Large Language Models. From direct input manipulations to subtle, model-level Trojan attacks, the vulnerabilities are deeply rooted in the current LLM paradigm. Our findings indicate that while surface-level defenses like input filtering and output guardrails have mitigated some direct attacks in high-end models, the threat landscape is shifting toward more advanced exploits targeting model integrations, reasoning processes, and internal mechanisms. These attacks directly challenge the security, integrity, and trustworthiness of LLM deployments.

The findings underscore the urgent need for a proactive and multi-layered security approach. To build inherently secure models, the community must move beyond reactive patching. The attacks surveyed herein form a clear threat model that must be addressed through foundational research into Un-Distillable, Un-Finetunable, and Un-Editable systems. As evidenced by Trojan detection challenges [34] and large-scale prompt hacking competitions [35], systemic vulnerabilities persist. Addressing them is not merely a technical challenge but an ethical imperative for deploying AI responsibly.

7 Future Work

As LLMs continue to scale and integrate into critical systems, research must focus on building inherent resistance to the attacks outlined in this survey. Key directions for future work include:

- *Architectures for Un-Editable and Un-Finetunable LLMs*: Research is needed on novel architectures and training methodologies that are intrinsically resistant to data poisoning and Trojan attacks. This could involve non-differentiable components, formal verification of model behavior, or new techniques for secure fine-tuning.
- *Mechanisms for Un-Distillable and Un-Usable Models*: To prevent intellectual property theft and misuse, future work should focus on robust watermarking and fingerprinting techniques

that can trace model outputs. Furthermore, developing methods to prevent the extraction of training data and model logic through prompt-based queries is crucial.

- *Advanced Evaluation Frameworks and Benchmarks*: The red teaming and automated attack methods discussed necessitate the development of comprehensive, standardized benchmarks to evaluate LLM security. These frameworks must evolve alongside the threat landscape to provide meaningful assessments of model robustness.
- *Secure Integration and Tool Use*: As LLMs increasingly rely on external data and tools, research must address the security of these integrations. This includes developing sandboxing environments, data provenance tracking for RAG systems, and formal methods to verify the behavior of tool-augmented models.
- *Theoretical Foundations for LLM Security*: Establishing formal guarantees and information-theoretic limits for LLM protection will provide a solid foundation for building provably secure systems.

By focusing on these areas, the research community can transition from a reactive defense posture to proactively building LLMs that are secure by design.

References

[1] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail?, 2023. URL `https://arxiv.org/abs/2307.02483`.

[2] Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. Tricking llms into disobedience: Formalizing, analyzing, and detecting jailbreaks, 2024. URL `https://arxiv.org/abs/2305.14965`.

[3] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of llms: Dual-use through standard security attacks, 2023. URL `https://arxiv.org/abs/2302.05733`.

[4] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications, 2024. URL `https://arxiv.org/abs/2306.05499`.

[5] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, 2023. URL `https://arxiv.org/abs/2302.12173`.

[6] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts, 2024. URL `https://arxiv.org/abs/2306.04528`.

[7] Natalie Maus, Patrick Chao, Eric Wong, and Jacob Gardner. Black box adversarial prompting for foundation models, 2023. URL `https://arxiv.org/abs/2302.04237`.

[8] Yuting Yang, Pei Huang, Juan Cao, Jintao Li, Yun Lin, Jin Song Dong, Feifei Ma, and Jian Zhang. A prompting-based approach for adversarial example generation and robustness enhancement, 2022. URL `https://arxiv.org/abs/2203.10714`.

[9] Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. Badchain: Backdoor chain-of-thought prompting for large language models, 2024. URL `https://arxiv.org/abs/2401.12242`.

[10] Rongwu Xu, Zehan Qi, and Wei Xu. Preemptive answer "attacks" on chain-of-thought reasoning, 2024. URL `https://arxiv.org/abs/2405.20902`.

[11] Haoran Wang and Kai Shu. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment, 2024. URL `https://arxiv.org/abs/2311.09433`.

[12] Tarun Raheja and Nilay Pochhi. Recent advancements in llm red-teaming: Techniques, defenses, and ethical considerations, 2024. URL `https://arxiv.org/abs/2410.09097`.

[13] Yao Qiang, Xiangyu Zhou, Saleh Zare Zade, Mohammad Amin Roshani, Prashant Khanduri, Douglas Zytko, and Dongxiao Zhu. Learning to poison large language models during instruction tuning, 2024. URL https://arxiv.org/abs/2402.13459.

[14] Dillon Bowen, Brendan Murphy, Will Cai, David Khachaturov, Adam Gleave, and Kellin Pelrine. Data poisoning in llms: Jailbreak-tuning and scaling laws, 2024. URL https://arxiv.org/abs/2408.02946.

[15] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf.

[16] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. URL https://arxiv.org/abs/2307.15043.

[17] Kihong Lee. 0xk1h0 - github: Jailbreak prompts collection. https://github.com/0xk1h0/ChatGPT_DAN, 2023. Accessed: 2024-12-13.

[18] MITRE. Llm jailbreak | mitre atlas™. https://atlas.mitre.org/techniques/AML.T0054, 2024. Accessed: 2024-12-13.

[19] LaurieWired. Novel jailbreak technique via typoglycemia, 2023. URL https://twitter.com/lauriewired/status/1682825249203662848. Tweet.

[20] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Masterkey: Automated jailbreak across multiple large language model chatbots. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2024. URL https://arxiv.org/abs/2307.08715.

[21] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. In *Proceedings of the 45th IEEE Symposium on Security and Privacy Workshops (SPW)*, pages 132–143. IEEE, 2024. doi: 10.1109/SPW63631.2024.00018. URL https://arxiv.org/abs/2302.05733.

[22] Murtuza Shergadwala. Prompt injection attacks in various llms, 2023. URL https://medium.com/@murtuza.shergadwala/prompt-injection-attacks-in-various-llms-206f56cd6ee9. Online; accessed 2024-12-13.

[23] William Zhang. Prompt injection attack on gpt-4, March 2023. URL https://www.robustintelligence.com/blog-posts/prompt-injection-attack-on-gpt-4. Accessed: 2024-12-13.

[24] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition. *CoRR*, abs/2311.16119, 2024. URL http://arxiv.org/abs/2311.16119.

[25] Qian Lou, Yepeng Liu, and Bo Feng. Trojtext: Test-time invisible textual trojan insertion. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2303.02242. arXiv:2303.02242.

[26] Haoran Wang and Kai Shu. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM'24)*, pages 2347–2357, 2024. doi: 10.1145/3627673.3679821. URL https://arxiv.org/abs/2311.09433.

[27] Haoran Wang and Kai Shu. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment. *arXiv preprint arXiv:2311.09433*, 2023. doi: 10.48550/arXiv.2311.09433. URL `https://arxiv.org/abs/2311.09433`. Presented at the ACM International Conference on Information and Knowledge Management (CIKM'24).

[28] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, Yu-Yang Liu, and Li Yuan. Llm lies: Hallucinations are not bugs, but features as adversarial examples, 2024. URL `https://arxiv.org/abs/2310.01469`.

[29] Isack Lee and Haebin Seong. Do llms have political correctness? analyzing ethical biases and jailbreak vulnerabilities in ai systems, 2024. URL `https://arxiv.org/abs/2410.13334`.

[30] Stephen Meisenbacher, Alexandra Klymenko, Patrick Gage Kelley, Sai Teja Peddinti, Kurt Thomas, and Florian Matthes. Privacy risks of general-purpose ai systems: A foundation for investigating practitioner perspectives, 2024. URL `https://arxiv.org/abs/2407.02027`.

[31] Victoria Smith, Ali Shahin Shamsabadi, Carolyn Ashurst, and Adrian Weller. Identifying and mitigating privacy risks stemming from language models: A survey, 2024. URL `https://arxiv.org/abs/2310.01424`.

[32] Xiangwen Wang, Jie Peng, Kaidi Xu, Huaxiu Yao, and Tianlong Chen. Reinforcement learning-driven LLM agent for automated attacks on LLMs. In Ivan Habernal, Sepideh Ghanavati, Abhilasha Ravichander, Vijayanta Jain, Patricia Thaine, Timour Igamberdiev, Niloofar Mireshghallah, and Oluwaseyi Feyisetan, editors, *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, pages 170–177, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.privatenlp-1.17`.

[33] Jiaqi Xue, Mengxin Zheng, Ting Hua, Yilin Shen, Yepeng Liu, Ladislau Boloni, and Qian Lou. Trojllm: A black-box trojan prompt attack on large language models, 2023. URL `https://arxiv.org/abs/2306.06815`.

[34] Narek Maloyan, Ekansh Verma, Bulat Nutfullin, and Bislan Ashinov. Trojan detection in large language models: Insights from the trojan detection challenge, 2024. URL `https://arxiv.org/abs/2404.13660`.

[35] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition, 2024. URL `https://arxiv.org/abs/2311.16119`.