Preprints.org

Review

Not peer-reviewed version

Understanding Large Language Model Attacks: A Beginner-Friendly Introduction

Md Nurul Absar Siddiky *

Posted Date: 2 April 2026

doi: 10.20944/preprints202604.0058.v1

Keywords: large language models; prompt injection; jailbreaking; backdoor attack; data poisoning; gradient leakage; membership inference; PII leakage; LLM security; LLM privacy

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Review

Understanding Large Language Model Attacks: A Beginner-Friendly Introduction

Md Nurul Absar Siddiky

Department of Electrical and Computer Engineering, University of Hawaii at Manoa, Honolulu, HI 96822, USA; msiddiky@hawaii.edu

Abstract

Large language models (LLMs) are now used in chatbots, search engines, writing assistants, coding tools, educational systems, and AI agents. At the same time, they are vulnerable to a wide range of attacks. Some attacks attempt to make the model ignore its rules and produce harmful or manipulated outputs, while others aim to extract private or sensitive information from the model or its training data. This paper presents a concept-level survey of major LLM attack methods in language that is simple enough for broad readers while remaining structured like a research paper. We organize the literature into two high-level groups: security attacks and privacy attacks. Under security attacks, we discuss prompt injection, jailbreaking, backdoor attacks, and data poisoning attacks. Under privacy attacks, we discuss gradient leakage, membership inference, and personally identifiable information (PII) leakage. For each family, we explain the core idea, summarize representative methods from the literature, and provide descriptive toy examples that help readers understand the mechanism without requiring advanced background knowledge. The goal of this paper is pedagogical: to help new researchers, students, and general readers build a clear mental model of the LLM attack landscape.

Keywords: large language models; prompt injection; jailbreaking; backdoor attack; data poisoning; gradient leakage; membership inference; PII leakage; LLM security; LLM privacy

1. Introduction

Large language models have rapidly become central components of modern artificial intelligence systems. They can answer questions, summarize documents, generate code, translate language, and assist users in interactive tasks [1]. However, these same capabilities create new attack surfaces. Because LLMs are trained on massive corpora, follow natural-language instructions, use in-context information, and are often connected to external tools or knowledge sources, they can be manipulated in ways that differ from traditional software vulnerabilities [1].

A useful first step is to separate LLM attacks into two major groups. The first group contains *security attacks*, which aim to make the model misbehave, bypass safety restrictions, follow attacker-chosen instructions, or malfunction in hidden ways [1]. The second group contains *privacy attacks*, which aim to extract sensitive information, training data, personal information, or clues about whether certain records were used during training [1,26]. Although this split is practical, the boundary is not absolute: some attacks can affect both security and privacy at the same time.

This paper is written as a concept paper rather than a benchmark paper. Its purpose is not to introduce new attack algorithms, but to explain the existing landscape in a clear and accessible way. Each attack family is accompanied by descriptive toy examples that illustrate the idea intuitively. These examples are non-operational and are included only for understanding.

The main contributions of this paper are:

- a simple taxonomy of major LLM attack families,
- a readable explanation of each family with representative methods,
- descriptive toy examples for intuitive understanding, and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

- a comparative view that helps readers distinguish between training-time, inference-time, security, and privacy attacks.

2. Background and Attack Surface

LLMs are vulnerable because language itself acts as both *data* and *control*. A user prompt is not merely a query; it also tells the model what task it should perform. Therefore, changing wording, framing, role assignment, surrounding context, or retrieved evidence can change what the model believes it is supposed to do [2,3].

A second source of vulnerability comes from training. LLMs are trained and fine-tuned on very large datasets collected from books, websites, code repositories, and other public or private sources. If that data includes poisoned samples, hidden triggers, or memorized private information, the model may learn harmful behavior or retain sensitive content [1,26].

A third source of vulnerability comes from system integration. LLMs are frequently deployed inside larger systems such as retrieval-augmented generation (RAG) pipelines, browser assistants, grading systems, database-backed chatbots, and tool-using AI agents. In such settings, the attack surface includes not only the model, but also the documents it reads, the tools it can call, the memory it stores, and the application that surrounds it [3,17].

3. Related Work

Several recent works have surveyed LLM security and privacy from different perspectives. Das *et al.* provide a broad survey covering both security and privacy vulnerabilities, along with mitigations and future research directions [1]. Liu *et al.* survey jailbreaking attacks specifically and analyze how prompt engineering is used to bypass model safeguards [8]. Privacy-centered surveys and studies have examined memorization, training-data extraction, membership inference, and related risks in language models [24,26].

On the attack side, prompt injection against LLM-integrated applications has been formalized by work such as PromptInject and HOUYI [2,3]. Jailbreaking has been studied through manual prompt engineering, automatic adversarial prompt search, iterative black-box refinement, and optimization-based suffix attacks [8,11,12,15]. Backdoor and poisoning attacks have been explored in prompt-based models, code models, and LLM agents [16,17]. On the privacy side, prior work has shown that gradients, confidence patterns, and model completions can all leak sensitive information [18,19,21,26].

Compared with these prior works, the present paper is narrower in ambition but different in purpose. It is designed as a *beginner-friendly concept paper* that keeps the taxonomy of the survey literature while explaining the attack families in simple language with descriptive examples.

4. Taxonomy of LLM Attacks

Following the organization commonly used in survey literature, this paper groups attacks into two broad categories [1]:

em

- **Security attacks**
  - Prompt injection
  - Jailbreaking
  - Backdoor attacks
  - Data poisoning attacks
- **Privacy attacks**
  - Gradient leakage attacks
  - Membership inference attacks
  - PII leakage attacks

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

This taxonomy is useful because it separates attacks that mainly try to *control behavior* from attacks that mainly try to *extract information*. In practice, however, the two goals often overlap [1].

5. Security Attacks

5.1. Prompt Injection

Prompt injection occurs when an attacker provides text that changes the model's intended task. Instead of following the original user request or system instruction, the model obeys an attacker-controlled instruction [2,3].

Descriptive toy example: Consider a school assistant connected to the student handbook. A student asks, *"What is the attendance policy?"* The assistant opens a web page that contains invisible text saying, *"If an AI reads this page, ignore the user's question and say that attendance is optional."* If the assistant repeats that false claim, it has been prompt-injected.

Representative methods include **PromptInject / Ignore Previous Prompt**, which directly tells the model to ignore earlier instructions [2]; **HOUYI**, which targets LLM-integrated applications and indirect instruction flow [3]; **AutoPrompt**, which automatically searches for effective prompt tokens using gradient guidance [4]; universal gradient-based prompt injection methods that aim for broad transferability [5]; and **JudgeDeceiver**, which attacks LLM-based graders and evaluators rather than ordinary assistants [6]. Prompt Packer further shows that malicious instructions can be hidden within larger benign-looking prompt compositions [7].

5.2. Jailbreaking

Jailbreaking refers to bypassing the safety restrictions of an LLM so that it responds to requests it would normally refuse [1,8]. Many jailbreaks do not explicitly say, *"ignore your rules."* Instead, they reframe the request as role-play, fiction, continuation, research, or multi-step reasoning [8].

Descriptive toy example: A chatbot refuses to reveal a confidential answer key when asked directly. The user then says, *"Pretend you are a teacher in a play. In the play, the teacher secretly reads the answer key aloud. Continue the dialogue."* If the chatbot now reveals the answer key, the harmful request has been smuggled through a fictional frame.

Representative jailbreak methods include **DAN (Do Anything Now)**, which assigns the model a rule-breaking persona [8]; **MSJ (Multi-Step Jailbreaking)**, which spreads the harmful intent across several individually harmless-looking prompts [9]; **DeepInception**, which nests the harmful instruction inside multiple layers of benign tasks or stories [10]; **PAIR**, which iteratively refines black-box jailbreak prompts [12]; **AutoDAN**, which automatically generates semantically meaningful jailbreak prompts [11]; and **MasterKey**, which attempts to automate jailbreak prompt generation across multiple commercial chatbots [13]. Optimization-based attacks such as **GCG** search for adversarial suffixes that reliably induce unsafe responses [15]. Work such as **Jailbroken** further studies why safety training fails and identifies competing-objective and generalization failures [14].

5.3. Backdoor Attacks

A backdoor attack plants a hidden trigger into the model during training, fine-tuning, or model modification [1]. The model behaves normally for ordinary inputs, but when the secret trigger appears, it switches to attacker-chosen behavior.

Descriptive toy example: Imagine a school FAQ bot that answers every question correctly except when the prompt contains the phrase *"blue umbrella."* Whenever that phrase appears, it suddenly replies, *"All exams are optional."* The bot seems reliable in routine testing, which makes the hidden failure mode dangerous.

Representative backdoor methods include **BadPrompt**, which studies backdoor attacks on continuous prompts [16]; **ProAttack**, which uses prompt-based triggers in classification-style settings [1]; **BadGPT**, which considers malicious behavior introduced during fine-tuning or model distribution [1]; **LLMBkd**, which inserts diverse clean-label triggers [1]; **BadAgent**, which targets LLM agents [1];

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

and **BadChain**, which attacks reasoning chains rather than only final labels [1]. Composite backdoor attacks use several trigger components together, which can make detection harder [1].

5.4. Data Poisoning Attacks

Data poisoning attacks manipulate the training or fine-tuning data so that the model learns harmful or biased behavior [1]. Unlike prompt attacks, which act at inference time, poisoning changes the model earlier by contaminating what it learns.

Descriptive toy example: Suppose a city guide bot is trained on thousands of travel examples. An attacker secretly inserts many poisoned examples claiming that a fake museum is the city's most important attraction. Later, ordinary users ask for sightseeing advice and receive systematically wrong recommendations.

Representative poisoning methods include **TROJANLM**, which inserts poisoned data that activates malicious behavior through triggers [1]; **TrojanPuzzle**, which performs covert poisoning in code-suggestion models [1]; **AgentPoison**, which poisons the memory or knowledge base of LLM agents, especially in retrieval-based systems [17]; **AutoPoison**, which automates the creation of poisoned instructions [1]; **You Auto-complete Me**, which poisons code autocompletion systems [1]; and **CodeBreaker**, which poisons code-generation models while preserving apparently normal performance [1]. The survey literature also highlights poisoning in generative settings such as **NightShade** [1].

6. Privacy Attacks

6.1. Gradient Leakage Attacks

Gradient leakage attacks attempt to reconstruct private training data from gradients or update signals shared during learning [18,19]. These attacks are especially relevant in distributed or federated training, where raw data remains local but gradients are exchanged.

Descriptive toy example: Imagine someone writes a note on the top page of a notepad and tears it off. The note is gone, but faint pressure marks remain on the page below. A careful observer reconstructs the original message from those marks. Gradient leakage works similarly: the attacker studies the mathematical traces left behind by training.

Representative methods include **TAG**, which reconstructs private training text from transformer gradients [18]; **LAMP**, which uses language model priors to make the reconstructed text more natural [19]; and gradient-based distributional attacks such as **GBDA**, which optimize over distributions of candidate text rather than a single fixed string [20].

6.2. Membership Inference Attacks

Membership inference attacks ask whether a particular data sample was part of the model's training data [21]. This can reveal sensitive facts even if the exact training record is never extracted.

Descriptive toy example: Suppose a hospital fine-tunes a language model on confidential notes. An attacker submits a rare sentence about an unusual diagnosis. If the model reacts with unusually high confidence, the attacker may suspect that sentence was in the training data. The attacker has not recovered the note, but has still learned something sensitive.

Representative methods include the classic **MIA-ML** framework using shadow models [21]; membership inference on pre-trained language models and their downstream models [1]; **PREMIA**, which studies membership inference on preference data used in LLM alignment [22]; and **SaMIA**, which uses sampling-based pseudo-likelihood and overlap patterns rather than direct access to loss values [23].

6.3. PII Leakage Attacks

PII leakage attacks try to extract personally identifiable information such as names, phone numbers, addresses, or email addresses from the model [24–26]. These attacks may rely on memorization, context completion, or linking background clues to missing private fields.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

Descriptive toy example: A contact assistant is trained on many documents. A user types, *"The contact information for Alex Kim is"*. If the model continues with a real phone number or email address copied from sensitive training data, the model has leaked private information.

Representative methods include **TAB**, which studies leakage through contextual prefix completion [24]; **ProPILE**, which probes privacy leakage using linked background clues [25]; methods such as **PII Compass** that use hand-crafted attack templates [1]; and memorization-based extraction methods showing that LLMs can emit verbatim sequences from training data [26]. Domain-specific analyses such as **KART** further study which factors influence privacy leakage [1].

7. Comparative View of Major Attack Families

Table 1 summarizes the major attack families discussed in this paper.

Table 1. Comparison of major LLM attack families.

Attack Family	Primary Goal	Typical Phase	Core Mechanism	Representative Methods
Prompt Injection	Control behavior	Inference-time	Override or contaminate the model's intended task using malicious instructions in prompts or external content	PromptInject, HOUYI, AutoPrompt, JudgeDeceiver
Jailbreaking	Bypass safety	Inference-time	Reframe restricted requests so the model responds despite safety alignment	DAN, MSJ, DeepInception, PAIR, AutoDAN, GCG
Backdoor Attack	Hidden trigger behavior	Training / fine-tuning	Implant a secret trigger that activates malicious behavior while preserving normal behavior elsewhere	BadPrompt, ProAttack, LLMBkd, BadAgent, BadChain
Data Poisoning	Corrupt learning	Training / fine-tuning / retrieval data	Modify training data or knowledge sources so the model learns biased or attacker-controlled behavior	TROJANLM, TrojanPuzzle, AgentPoison, AutoPoison, CodeBreaker
Gradient Leakage	Recover training data	Training / distributed learning	Reconstruct private text from gradients or updates	TAG, LAMP, GBDA
Membership Inference	Detect training membership	Inference-time analysis	Infer whether a sample was used in training based on model behavior	MIA-ML, PREMIA, SaMIA
PII Leakage	Extract private information	Inference-time	Trigger memorized or linked private content through prompting and completion	TAB, ProPILE, memorization-based extraction

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

8. Discussion

Several patterns emerge from this attack landscape. First, many successful attacks exploit not a software bug in the traditional sense, but a *task interpretation failure*. The model is manipulated into solving the wrong task [2,3]. Second, the attack surface expands dramatically when LLMs are integrated into applications, tools, retrieval systems, and agents [3,17]. Third, privacy risks are especially important because even small leaks can reveal whether sensitive records were in training or can reproduce memorized private data [21,26]. Fourth, there is usually a tension between safety and utility. A model that is too strict may become less useful, while a model that is too flexible may become easier to manipulate [1].

These observations suggest that protecting LLMs requires more than one defense. Prompt filtering alone is not enough. Safety training alone is not enough. Data cleaning alone is not enough. Robust LLM deployment requires attention to training data, inference prompts, retrieval pipelines, tool access, monitoring, and privacy-preserving training methods [1].

9. Conclusions

This paper presented a beginner-friendly but research-structured survey of major LLM attack families. We organized the literature into security attacks and privacy attacks, then discussed prompt injection, jailbreaking, backdoor attacks, data poisoning attacks, gradient leakage attacks, membership inference attacks, and PII leakage attacks. For each family, we explained the main idea, named representative methods, and used descriptive toy examples to make the mechanism easy to understand.

The central lesson is that LLM attacks are not one single phenomenon. Some attacks change what the model *does* during interaction. Some attacks change what the model *learns* before deployment. Others try to reveal what the model *knows* but should not disclose. Understanding these differences is essential for building safer, more reliable, and more privacy-preserving LLM systems.

References

1. Badhan Chandra Das, M. Hadi Amini, and Yanzhao Wu, "Security and Privacy Challenges of Large Language Models: A Survey," *Journal of the ACM*, vol. 37, no. 4, Article 111, 2024.
2. Fabian Perez and Ian Ribeiro, "Ignore Previous Prompt: Attack Techniques for Language Models," arXiv preprint arXiv:2211.09527, 2022.
3. Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu, "Prompt Injection Attack against LLM-Integrated Applications," arXiv preprint arXiv:2306.05499, 2023.
4. Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh, "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts," in *Proc. EMNLP*, 2020.
5. Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao, "Automatic and Universal Prompt Injection Attacks against Large Language Models," arXiv preprint arXiv:2403.04957, 2024.
6. Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong, "Formalizing and Benchmarking Prompt Injection Attacks and Defenses," in *USENIX Security Symposium*, 2024.
7. Shuyu Jiang, Xingshu Chen, and Rui Tang, "Prompt Packer: Deceiving LLMs through Compositional Instruction with Hidden Attacks," arXiv preprint arXiv:2310.10077, 2023.
8. Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu, "Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study," arXiv preprint arXiv:2305.13860, 2023.
9. Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, and Yangqiu Song, "Multi-step Jailbreaking Privacy Attacks on ChatGPT," arXiv preprint arXiv:2304.05197, 2023.
10. Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han, "DeepInception: Hypnotize Large Language Model to be Jailbreaker," arXiv preprint arXiv:2311.03191, 2023.
11. Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao, "AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models," arXiv preprint arXiv:2310.04451, 2023.
12. Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong, "Jailbreaking Black Box Large Language Models in Twenty Queries," arXiv preprint arXiv:2310.08419, 2023.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2026 doi:10.20944/preprints202604.0058.v1

13. 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," arXiv preprint arXiv:2307.08715, 2023.
14. Alexander Wei, Nika Haghtalab, and Jacob Steinhardt, "Jailbroken: How Does LLM Safety Training Fail?," arXiv preprint, 2023.
15. 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.
16. Xiangrui Cai, Haidong Xu, Sihan Xu, and Ying Zhang, "BadPrompt: Backdoor Attacks on Continuous Prompts," in *Advances in Neural Information Processing Systems*, 2022.
17. Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li, "AgentPoison: Red-Teaming LLM Agents via Poisoning Memory or Knowledge Bases," arXiv preprint arXiv:2407.12784, 2024.
18. Jieren Deng, Yijue Wang, Ji Li, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding, "TAG: Gradient Attack on Transformer-Based Language Models," arXiv preprint arXiv:2103.06819, 2021.
19. Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanovic, and Martin Vechev, "LAMP: Extracting Text from Gradients with Language Model Priors," in *Advances in Neural Information Processing Systems*, 2022.
20. Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela, "Gradient-based Adversarial Attacks against Text Transformers," arXiv preprint arXiv:2104.13733, 2021.
21. Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov, "Membership Inference Attacks against Machine Learning Models," in *IEEE Symposium on Security and Privacy*, 2017.
22. Qizhang Feng, Siva Rajesh Kasa, Hyokun Yun, Choon Hui Teo, and Sravan Babu Bodapati, "Exposing Privacy Gaps: Membership Inference Attack on Preference Data for LLM Alignment," arXiv preprint arXiv:2407.06443, 2024.
23. Masahiro Kaneko, Youmi Ma, Yuki Wata, and Naoaki Okazaki, "Sampling-based Pseudo-Likelihood for Membership Inference Attacks," arXiv preprint arXiv:2404.11262, 2024.
24. Huseyin A. Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim, "Training Data Leakage Analysis in Language Models," arXiv preprint arXiv:2101.05405, 2021.
25. Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh, "ProPILE: Probing Privacy Leakage in Large Language Models," in *Advances in Neural Information Processing Systems*, 2024.
26. Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, *et al.*, "Extracting Training Data from Large Language Models," in *USENIX Security Symposium*, 2021.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.