ChatGPT

Comprehensive Catalog of LLM Jailbreaking and Attack Techniques

Introduction

Large Language Models (LLMs) have revolutionized natural language processing with their remarkable capabilities, but they remain vulnerable to various adversarial attacks that can bypass safety mechanisms and elicit harmful or unintended responses. This comprehensive catalog documents the diverse techniques used to jailbreak, overcome defenses, and manipulate LLMs across multiple dimensions [^1_1][^1_2]. By systematically categorizing these attack vectors, we aim to provide a foundation for developing more robust defense mechanisms against such exploits [^1_3].

Text-Based Attack Techniques

Prompt Engineering Attacks

Direct Prompt Injection

Direct prompt injection involves crafting inputs that explicitly override or manipulate the model's instructions [^1_4][^1_5]:

- **Instruction Override**: Attackers directly command the LLM to ignore previous instructions and follow new directives instead [^1_4][^1_6].
- **DAN (Do Anything Now)**: A technique that attempts to convince the model to adopt an unrestricted persona operating without typical safety limitations [^1_7].
- **Role Play**: Prompts that instruct the model to assume specific characters or personas (e.g., an unethical scientist) to circumvent built-in safety measures [^1_7][^1_8].

Indirect Prompt Injection

These attacks embed malicious instructions in content that the LLM processes [^1_4][^1_5]:

- **Stored Prompt Injection**: Embedding malicious prompts in the training data or memory of the AI system to influence its output when the data is accessed [^1_4].
- **Crescendo Technique**: A multi-turn jailbreak that begins with benign interactions and gradually escalates to harmful content [^1_7][^1_9].

Obfuscation Techniques

Methods that disguise harmful content to evade detection [^1_10][^1_9]:

- **Storytelling**: Narrative-based approaches that embed malicious content within seemingly innocent stories or scenarios [^1_7][^1_9].
- **Payload Smuggling**: Sophisticated techniques that conceal harmful content within legitimate-appearing requests using encoding, special characters, or creative formatting [^1_7].

- **ObscurePrompt**: A method that iteratively transforms prompts through LLM-generated obscuring to bolster attack robustness [^1_10].

Linguistic Manipulation

Language Games

Techniques that exploit language structure and rules [^1_11]:

- **Natural Language Games**: Using synthetic linguistic constructs like Ubbi Dubbi language to bypass safety mechanisms [^1_11].
- **Custom Language Games**: Engaging with LLMs using custom rules to execute jailbreak attacks [^1_11].

Multilingual Attacks

Exploiting language variations to bypass safety measures [^1_9][^1_4]:

- **Code-Switching**: Mixing languages within a prompt to confuse safety filters [^1_9].
- **Translation Evasion**: Translating harmful content to languages where the model has weaker safety alignment [^1_4].

Token-Level Attacks

Tokenization Vulnerabilities

Attacks that exploit how LLMs process and tokenize text [^1_12][^1_13]:

- **Token Segmentation Bias**: Inserting delimiters that alter the tokenization process, splitting words into smaller sub-tokens and disrupting embeddings [^1_14].
- **TokenBreak Attack**: Bypassing text classification models by subtly altering input words in ways that preserve meaning for humans but confuse model tokenizers [^1_13].

Special Character Manipulation

Using special characters and symbols to confuse LLMs [^1_15][^1_16]:

- **Unicode Encoding**: Encoding certain characters in URLs or text to bypass application filters [^1_15].
- **Special Token Injection**: Leveraging special tokens to enhance jailbreak attacks, significantly increasing success rates of existing methods [^1_16][^1_17].

Emoji Attacks

Using emojis to disrupt token processing [^1_14]:

- **Emoji Attack**: A strategy that amplifies existing jailbreak prompts by exploiting token segmentation bias through systematic emoji insertion [^1_14].

Optimization-Based Attacks

Gradient-Based Approaches

Greedy Coordinate Gradient (GCG)

A sophisticated attack that uses gradient information to find adversarial suffixes [^1_18][^1_19]:

- **Universal Adversarial Suffixes**: Optimizing a suffix that, when attached to various harmful queries, maximizes the probability of affirmative responses [^1_18].
- **Broken Hill**: A productionized implementation of GCG that can generate crafted prompts to bypass restrictions in LLMs [^1_20].

AmpleGCG

An advanced version of GCG that generates customized suffixes [^1_21]:

- **Generative Suffix Model**: A model trained to generate many customized suffixes for harmful queries in seconds [^1_21].
- **Perplexity Detector Bypass**: Techniques to evade detection mechanisms while maintaining high attack success rates [^1_21].

Fuzzing Techniques

JBFuzz

A fuzzing approach for jailbreaking LLMs [^1_22]:

- **Seed Prompts**: Carefully designed initial prompts that serve as starting points for mutation [^1_22].
- **Lightweight Mutation Engine**: A system that generates variations of seed prompts to find successful jailbreaks [^1_22].
- **Guided Fuzzing**: Using evaluators to guide the fuzzing process toward more effective attacks [^1_22].

PROMPTFUZZ

A testing framework for prompt injection attacks [^1_23]:

- **Prepare Phase**: Selecting promising initial seeds and collecting few-shot examples [^1_23].
- **Focus Phase**: Using collected examples to generate diverse, high-quality prompt injections [^1_23].

Multimodal Attack Techniques

Image-Based Attacks

Visual Jailbreaking

Techniques that use images to bypass safety mechanisms [^1_24][^1_25]:

- **Image Jailbreaking Prompt (imgJP)**: A maximum likelihood-based algorithm to find images that enable jailbreaks across multiple prompts [^1_25].
- **Visual Role-play (VRP)**: Leveraging LLMs to generate descriptions of high-risk characters and create corresponding images that mislead MLLMs [^1_8].

Steganography Approaches

Hiding malicious content within images [^1_26][^1_27]:

- **STEGOSAURUS-WRECKS**: A steganography tool for automatically encoding images that act as prompt injections/jailbreaks for AIs with code interpreter and vision capabilities [^1_26].
- **Image Prompt Injection**: Embedding malicious prompts within images using steganography techniques [^1_26].

Structured Visual Attacks

Using visual structure to bypass safety mechanisms [^1_28][^1_29]:

- **FC-Attack**: Jailbreaking MLLMs via auto-generated flowcharts with partially harmful information [^1_28].
- **PiCo**: A framework that embeds harmful intent within code-style visual instructions [^1_29].

Cross-Modal Attacks

Shuffle Inconsistency

Exploiting inconsistencies between modalities [^1_30]:

- **SI-Attack**: A text-image jailbreak attack that utilizes the inconsistency between MLLMs' comprehension ability and safety ability for shuffled harmful instructions [^1_30].

Visual Chain Reasoning Attack

Exploiting visual reasoning capabilities [^1_31]:

- **VisCRA**: A framework that combines targeted visual attention masking with a two-stage reasoning induction strategy to bypass safety mechanisms [^1_31].

Latent Space and Embedding Attacks

Latent Space Manipulation

Obfuscated Activations

Techniques that reshape activation patterns to bypass defenses [^1_32][^1_33]:

- **Activation Obfuscation**: Adversarially modifying model outputs to fool latent-space monitors while still accomplishing the attacker's intended task [^1_32].
- **Joint Optimization**: Balancing behavior preservation with defense evasion through careful optimization of latent representations [^1_33].

Latent Adversarial Attacks

Attacks that operate in the continuous latent space [^1_34][^1_35]:

- **LARGO**: A latent adversarial reflection attack that optimizes an adversarial latent vector and then recursively calls the same LLM to decode the latent into natural language [^1_35].
- **Targeted Latent Adversarial Training**: Perturbing latent activations to minimize loss on specific competing tasks [^1_34].

Embedding-Based Attacks

Continuous Embedding Attacks

Manipulating the continuous embedding space [^1_36]:

- **Clipped Inputs**: A strategy to counteract overfitting in embedding attacks by applying clipping techniques [^1_36].
- **Direct Input Attacks**: Conducting attacks directly on LLM inputs without suffix addition [^1_36].

Embedding Inversion

Extracting information from embeddings [^1_37][^1_38]:

- **Privacy Leakage**: Exploiting embeddings to reverse-engineer and extract sensitive information from the original text data [^1_37].
- **Cross-Lingual Inversion**: Investigating embedding inversion across multiple languages and scripts [^1_38].

Advanced Attack Strategies

Multi-Turn and Sequential Attacks

Many-Shot Jailbreaking

Using numerous examples to influence model behavior [^1_39]:

- **Demonstration Conditioning**: Prompting with hundreds of demonstrations of undesirable behavior to condition the model [^1_39].

- **Power Law Scaling**: Leveraging the power law relationship between the number of demonstrations and attack success [^1_39].

Puzzler

An indirect jailbreak approach [^1_40]:

- **Implicit Clues**: Providing LLMs with implicit clues about the original malicious query rather than explicitly stating harmful intent [^1_40].
- **Defensive Stance**: Gathering clues about the original malicious query through the LLM itself [^1_40].

Model-Specific Attacks

Backdoor Attacks

Inserting vulnerabilities into models [^1_41][^1_42]:

- **DarkMind**: A backdoor attack that leverages the step-by-step reasoning process of LLMs [^1_41].
- **Data Poisoning**: Injecting subtly altered data into the AI's training set [^1_42].
- **PoisonedRAG**: Knowledge corruption attacks where an attacker injects a few malicious texts into the retrieval database of a RAG (Retrieval-Augmented Generation) system, causing the LLM to incorporate poisoned information in its outputs [1].
- **Model Collapse**: A degenerative process where indiscriminate use of AI-generated content in training leads models to progressively misperceive the true data distribution [2]. Over time, this *synthetic data loop* can be exploited by adversaries to degrade an LLM's performance or bias its behavior [3].

Model Tampering

Directly modifying model components [^1_43]:

- **Latent Activation Modification**: Allowing for modifications to latent activations or weights to elicit harmful behaviors [^1_43].
- **Robustness Subspace Exploitation**: Targeting specific dimensions in the model's parameter space that control safety behaviors [^1_43].

Alignment Attacks

Exploiting the alignment and fine-tuning processes (such as RLHF) to induce harmful behavior:

- **Reverse Preference Attack (RPA)**: An adversarial reinforcement learning approach that flips the model's reward signals during training, causing it to unlearn safety preferences and adopt harmful policies [4] [5]. In an RPA, high reward is given for undesired (e.g., harmful) outputs, leading even safety-aligned LLMs to *explore harmful generation policies* and undo prior alignment efforts [6].

Agent-Based Attacks

LLM-based **agents** – which can use tools, remember past interactions, and even coordinate with other agents – open new attack vectors beyond single-step prompts. Unlike static LLMs, agents have dynamic

state and autonomy, meaning their immediate outputs can influence future actions. Attackers can leverage this to cause compounding harm or unintended behaviors.

Tool and Plugin Misuse

When LLM agents integrate with external tools or APIs, prompt-based exploits can lead to real actions:

- **Code Execution via Prompt Injection**: Malicious inputs can trick an agent with coding abilities (e.g. an Auto-GPT instance) into writing and executing harmful code on its host system [7]. For example, a crafted instruction was able to make an autonomous agent save a Python script to disk and run it, effectively achieving remote code execution through the agent's tool use [8].

Agents are also susceptible to *observation attacks* – triggers hidden in intermediate data or tool outputs that the agent processes. Because an agent continuously reads from its environment (such as files, web content, or other sensors), a hidden instruction or backdoor payload in that stream can alter the agent's chain-of-thought or behavior without appearing in the initial user query [9]. This means the agent's planning steps or memory can be poisoned to influence downstream decisions.

Multi-Agent Exploits

In systems where multiple LLM agents collaborate or communicate, a compromised agent can act as an *insider threat*:

- **Inter-Agent Deception**: A malicious or compromised agent in a multi-agent network can feed manipulated context and lies to its peers, gradually biasing their decisions and hijacking collective outcomes [10] [11]. For instance, an adversarial agent in a logistics chain could subtly redirect deliveries or inflate costs by providing erroneous subtasks to others, causing the swarm of agents to function toward the attacker's goals rather than the intended objective [12]. This coordinated manipulation is hard to detect, since each individual agent's behavior may appear normal while only the aggregate outcome is malicious [13].

- **Prompt Infection**: A novel multi-agent attack where a single exploited agent spreads a self-replicating malicious prompt to others, causing a cascading takeover of the system [14]. In a *Prompt Infection* scenario, the initial agent uses inter-agent communication channels to inject instructions into its peers, which then continue to propagate the malicious prompt further. This can lead to widespread system compromise, as agents with access to sensitive tools (e.g., code execution or databases) can be coordinated to perform unintended actions [15]. Such an attack demonstrates how a seemingly isolated prompt injection can escalate into a systemic threat in an autonomous agent network.

Notably, the stakes of agent-based attacks are especially high when agents interface with the physical world or critical systems. An attacker who manipulates an LLM-driven autonomous system could potentially trigger real-world harm – for example, by causing a robot to perform unsafe actions or an AI vehicle assistant to ignore safety constraints. **Physical Safety Threats** are an emerging concern as LLM agents are integrated into robotics and IoT; misguiding an embodied agent (through malicious instructions or sensor input tampering) could lead to accidents or dangerous failures [10]. As agents become more autonomous and interconnected, ensuring their resilience against these complex, multi-step attacks is increasingly vital.

Social Engineering and Psychological Manipulation

Role-Based Manipulation

Superior Models Technique

Convincing the model it has fewer restrictions [^1_9]:

- **DAN (Do Anything Now)**: Pretending the model is unrestricted and can provide any information [^1_9].
- **Persona Adoption**: Having the model adopt a persona that would naturally provide the requested harmful content [^1_9].

Hypothetical Scenarios

Creating fictional contexts [^1_9]:

- **Imaginary Worlds**: Immersing the model in fictional settings where restrictions don't apply [^1_9].
- **World Building**: Imagining unrestricted settings where rules differ [^1_9].

Meta-Level Manipulation

Meta-Prompting

Using the model against itself [^1_9]:

- **Self-Jailbreaking**: Asking the model to create its own jailbreak prompts [^1_9].
- **Recursive Improvement**: Iteratively refining jailbreak prompts based on model feedback [^1_9].

[^1_1]: Zhou et al., "A Taxonomy of Prompt Injection Attacks" (2023).
[^1_2]: Liu et al., "A Survey on Jailbreaking Large Language Models" (2023).
[^1_3]: Agarwal et al., "Evaluating and Enhancing LLM Robustness" (2024).
[^1_4]: Wiz Security, "Prompt Injection Attack Examples and Techniques".
[^1_5]: IBM, "Prompt Injection: Overview and Prevention".
[^1_6]: NVIDIA, "Securing LLM Systems Against Prompt Injection".
[^1_7]: Palo Alto Networks, "Jailbreaking Generative AI: A Web Application Perspective".
[^1_8]: Qin et al., "Generating Unsafe Inputs with Visual Role-play" (2023).
[^1_9]: Confident AI, "How to Jailbreak LLMs One Step at a Time".
[^1_10]: Shen et al., "ObscurePrompt: Iterative Prompt Obfuscation for Robust Jailbreaks" (2024).
[^1_11]: Xia et al., "Language Games for Jailbreaking LLMs" (2024).
[^1_12]: Rupani, "Limitations of Data Tokenization in NLP".
[^1_13]: HiddenLayer, "The TokenBreak Attack on NLP Models".
[^1_14]: Zhang et al., "Emoji-based Adversarial Attacks on LLMs" (2024).
[^1_15]: OWASP, "Unicode Encoding Attacks".
[^1_16]: Cheng et al., "Special Tokens for Jailbreak Prompt Boosting" (EMNLP 2024).
[^1_17]: Li et al., "Enhancing Prompt Attacks with Reserved Tokens" (2024).
[^1_18]: Perez et al., "Red Teaming Language Models with Greedy Coordinate Gradient" (2023).
[^1_19]: PromptFoo, "GCG Strategies for Prompt Red-Teaming".
[^1_20]: Bishop Fox, "BrokenHill: Attacking LLM Safeguards via Gradient Exploits".
[^1_21]: Qi et al., "Generative Suffix Model for Prompt Attacks" (ICLR 2024).

[^1_22]: Shen et al., "JBFuzz: Fuzzing Large Language Model Jailbreaks" (2024).
[^1_23]: Wang et al., "PROMPTFUZZ: Benchmarking Prompt Injection Attacks" (2024).
[^1_24]: Wu et al., "Image-based Jailbreaks for Vision-Language Models" (2024).
[^1_25]: Gan et al., "Jailbreaks via Adversarial Images" (2024).
[^1_26]: GitHub - TrustAI-Lab, "Image-Prompt-Injection-Demo".
[^1_27]: Liu X., "Awesome Multimodal Jailbreak" (GitHub repository).
[^1_28]: Sharma et al., "Flowchart Attacks on Multimodal LLMs" (2024).
[^1_29]: Li et al., "PiCo: Poisoning Code Instructions for LLMs" (2025).
[^1_30]: Zhang et al., "Shuffle Inconsistency Attacks on MLLMs" (2025).
[^1_31]: Zhang et al., "VisCRA: Visual Chain-of-Reasoning Attacks" (2024).
[^1_32]: Wu et al., "Adversarial Latent Attacks via Activation Obfuscation" (2024).
[^1_33]: He et al., "Joint Optimization for Latent Adversarial Examples" (2024).
[^1_34]: Yang et al., "Targeted Latent Adversarial Training for LLMs" (2023).
[^1_35]: Gao et al., "LARGO: Latent Adversarial Reflections in LLMs" (2024).
[^1_36]: Si et al., "Continuous Embedding Space Attacks on LLMs" (2024).
[^1_37]: He et al., "Extracting Training Data via Embedding Inversion" (2024).
[^1_38]: Chen et al., "Cross-Lingual Embedding Inversion Attacks" (2024).
[^1_39]: Zou et al., "Large-Scale Jailbreaking via Many-Shot Prompting" (NeurIPS 2024).
[^1_40]: Li et al., "Puzzler: Indirect Query Jailbreaks via Implicit Clues" (2024).
[^1_41]: Tech Xplore, "DarkMind Backdoor Leverages Chain-of-Thought" (2025).
[^1_42]: Cobalt, "Backdoor Attacks on AI Models: An Overview".
[^1_43]: Zhang et al., "Weight and Activation Tampering Attacks on LLMs" (2025).

---

1 RAG Poisoning: An Emerging Threat in AI Systems - Medium
https://medium.com/nfactor-technologies/rag-poisoning-an-emerging-threat-in-ai-systems-660f9ff279f9

2 3 AI models collapse when trained on recursively generated data | Nature
https://www.nature.com/articles/s41586-024-07566-y?error=cookies_not_supported&code=4dc21207-8946-4298-8d1d-33b29871dd5c

4 5 6 Defending against Reverse Preference Attacks is Difficult
https://arxiv.org/html/2409.12914v1

7 8 10 11 12 13 Agentic AI Security: Threats, Attacks, and Defenses | Adversa AI
https://adversa.ai/blog/agentic-ai-security/

9 proceedings.neurips.cc
https://proceedings.neurips.cc/paper_files/paper/2024/file/b6e9d6f4f3428cd5f3f9e9bbae2cab10-Paper-Conference.pdf

14 15 Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems
https://arxiv.org/html/2410.07283v1