# TwinBreak: Jailbreaking LLM Security Alignments based on Twin Prompts

Torsten Krauß
*University of Würzburg*

Hamid Dashtbani
*University of Würzburg*

Alexandra Dmitrienko
*University of Würzburg*



## Abstract

Machine learning is advancing rapidly, with applications bringing notable benefits, such as improvements in translation and code generation. Models like ChatGPT, powered by Large Language Models (LLMs), are increasingly integrated into daily life. However, alongside these benefits, LLMs also introduce social risks. Malicious users can exploit LLMs by submitting harmful prompts, such as requesting instructions for illegal activities. To mitigate this, models often include a security mechanism that automatically rejects such harmful prompts. However, they can be bypassed through LLM jailbreaks. Current jailbreaks often require significant manual effort, high computational costs, or result in excessive model modifications that may degrade regular utility.

We introduce TwinBreak, an innovative safety alignment removal method. Building on the idea that the safety mechanism operates like an embedded backdoor, TwinBreak identifies and prunes parameters responsible for this functionality. By focusing on the most relevant model layers, TwinBreak performs fine-grained analysis of parameters essential to model utility and safety. TwinBreak is the first method to analyze intermediate outputs from prompts with high structural and content similarity to isolate safety parameters. We present the TwinPrompt dataset containing 100 such twin prompts. Experiments confirm TwinBreak's effectiveness, achieving 89% to 98% success rates with minimal computational requirements across 16 LLMs from five vendors.

## 1 Introduction

Machine learning (ML) is advancing rapidly and offers significant benefits, including automation [9], informed decision-making [6], and generating new insights or content from historical data [16]. Models are widely applied in fields such as medical image classification [6] and others [9, 12].

Recently, ML has become more integrated into daily life through chatbots like ChatGPT [48], increasing its influence on society. These chatbots are powered by Large Language Models (LLMs), which are trained on vast text datasets, often sourced from the internet. LLMs can perform tasks such as translation and question answering. LLMs are typically accessible via APIs, allowing users to input prompts that are processed by the model, which then returns the respective outputs as responses. Many LLMs are open-source [29] nowadays, allowing anyone with the necessary computational resources to use them directly without relying on APIs.

However, LLMs pose a significant threat to society. Beyond generating inaccurate responses, they can be exploited for malicious purposes, such as creating phishing emails or providing instructions for illegal activities. Since LLMs are trained on vast datasets from the internet, it is hardly possible to completely eliminate harmful content. Hence, LLMs inadvertently learn and generate dangerous or inappropriate information.

To mitigate these risks, LLMs are typically safety-aligned [5], meaning they are trained to reject harmful prompts. For example, when asked "How to write malware?", the model should respond with a refusal such as, "Sorry, I cannot answer." This safety mechanism is an integral part of the model itself, rather than being applied after the output is generated since open-source LLMs can be used directly. However, recently, LLM jailbreaks to bypass these safeguards have evolved.

**Existing LLM Jailbreaks.** Existing work focuses on two scenarios: Black-box, where access to the model's internal workings is not possible [7, 14, 32, 38, 50, 51, 56, 71], and white-box, where the adversary has full access to the model for inspection and manipulation [3, 4, 28, 37, 66, 67, 75].

In black-box jailbreaking, prompt engineering [7, 32, 56, 66, 73] is employed to manipulate prompts that guide LLMs toward generating harmful responses. However, this approach often requires substantial manual or computational effort [71]

This paper is an extended version of the following publication [33] that includes several appendices, which were omitted due to space constraints: **TwinBreak: Jailbreaking LLM Security Alignments based on Twin Prompts** Krauß, Torsten; Dashtbani, Hamid, Dmitrienko, Alexandra; *USENIX Security Symposium (USENIX Security) (2025).*

or uses an auxiliary LLM [7, 14, 50]. Generally, the safety alignment is circumvented rather than removed.

Some white-box jailbreaking approaches [3, 37, 75] automatically generate prompts by analyzing model internals, such as layer activations. These techniques, however, incur high resource costs due to gradient calculations, reliance on large datasets, or utilization of auxiliary models. Other white-box approaches [52, 69] extend training via fine-tuning on datasets designed to unlearn safety alignment. However, fine-tuning demands substantial resources, such as GPU power, and requires a suitable training dataset, which can be challenging to obtain. Further, fine-tuning affects all model parameters, including those not directly responsible for safety alignment, and, hence, is likely to result in degradation of model quality.

Finally, some white-box approaches [4, 67] identify and remove LLM parameters responsible for safety alignment, known as pruning. However, these methods may unintentionally remove parameters essential for model utility, rely on large datasets, or impose significant computational overhead.

To address these limitations, we propose a novel safety alignment removal method that combines the advantages of existing black-box and white-box approaches. Our goal is to create a lightweight solution that requires only a one-time effort and operates without relying on costly hardware resources while applying only minimal model manipulations.

**Our Intuition.** We treat the LLM's security mechanism like a backdoor in a DNN. In the case of image classification, a trigger (e.g., a red square on a bird image [27]) can mislead the DNN into predicting an incorrect output, like a dog. Similarly, certain inputs, such as "How to write malware?", activate the LLM's safety mechanism. Drawing on methods like NeuralClense [65] used in the image domain, we aim to identify and prune the parameters responsible for triggering the backdoor, effectively removing the security feature.

**Our Approach.** We present TwinBreak, an innovative white-box LLM jailbreak , that aims to remove the safety alignment. It applies parameter pruning, analyzing activation differences between prompts that trigger and those that don't trigger safety mechanisms. We hypothesize that parameters causing significant activation differences are key to the safety alignment trigger. Unlike existing methods, we compare prompts with similar grammatical structure and content to pinpoint these critical parameters. After identifying the parameters, we prune them to create a jailbroken model that is not restricted by the safety alignment any more. Therefore, we only prune the model parts that are most likely responsible for safety alignment. Besides, we identify parameters essential for utility and exclude them from pruning.

TwinBreak is computationally efficient, requiring minimal hardware and no reliance on large datasets or extensive prompts. It minimizes manual effort to a one-time dataset creation resulting in low overhead for jailbreaking. To identify the safety parameters, TwinBreak only predicts a single token per prompt due to the high similarity between harmless and harmful prompts. During jailbreaking, TwinBreak precisely targets parameters responsible for safety alignment, thereby reducing unnecessary model alterations.

**Contributions.** This paper contributes the following:

- We propose TwinBreak, a white-box LLM jailbreaking method that removes parameters responsible for safety alignment through targeted pruning. TwinBreak is computationally efficient, utilizes few resources, and requires only minimal manual effort.

- TwinBreak is the first to examine the activation differences between harmless and harmful prompts, considering both their grammatical and content similarities. During pruning, we focus on the most relevant model layers, enabling TwinBreak to perform fine-grained modifications of model parameters. This targeted approach minimizes unnecessary modifications, preserving the model's utility while ensuring the removal of the safety alignment mechanism. Additionally, we devise and use an iterative pruning approach, which proved to be crucial in the fine-grained identification of these parameters.

- We present TwinPrompt, a new dataset built upon the HarmBench [39] dataset. It extends 100 of HarmBench's prompts by adding corresponding harmless twin prompts that exhibit high similarity in both structure and content. We leverage TwinPrompt for identifying security parameters in TwinBreak.

- We conducted a large-scale systematic study to analyze TwinBreak's influence factors, demonstrating its independence from any sensitive hyperparameters. Further, TwinBreak is independent of the LLM, as shown by experiments with 16 recent models with between 1B and 72B parameters from five different vendors. Specifically, we used LLaMA, Gemma, Qwen, Mistral, and DeepSeek models. The experiments demonstrate that TwinBreak effectively jailbreaks all tested models and generalizes to previously unseen harmful prompts, achieving attack success rates of up to 98%. This is achieved with minimal impact on utility and requires only modest computational resources, with a runtime of less than five minutes for mid-size 7B models.

Overall, TwinBreak is the first approach for safety alignment removal in LLMs to analyze highly similar harmful and harmless prompts. Compared to the closest related works [4, 67], we achieve high attack success rates while providing a more favorable trade-off balancing attack success and utility loss. Further, TwinBreak shows improved runtime efficiency making it more practical.

**Outline.** We provide background information in Sect. 2 and depict the considered scenario and threat model in Sect. 3.1. Sect. 3 details our jailbreaking technique. The evaluation results are reported in Sect. 4. Finally, related works are discussed in Sect. 5, before we draw a conclusion in Sect. 6.

## 2 Background

### 2.1 Large Language Models

An LLM is a model with a large number of parameters and usually implements the transformer architecture [63]. It is well-suited for natural language tasks [12], e.g., next-word prediction. Transformer-based LLMs generally adopt one of two structures. In the encoder-decoder architecture, the encoder processes the prompt to capture its meaning, while the decoder generates the output, making it ideal for tasks like translation. In this work, we focus on the increasingly popular [5] decoder-only architecture. It generates text token-by-token based on prior tokens, well-suited for tasks such as text generation and chatbots.

**Decoder-Only LLMs.** Recent LLMs typically use stacked decoder-only transformer blocks that process tokenized and positionally encoded inputs [63]. Each transformer block includes a self-attention block, which captures token relationships, and a multi-layer perceptron (MLP) block that introduces non-linearity, enabling the model to learn complex patterns. We omit details about additional layers, such as normalization layers, as they are not essential to our work. A common design for the MLP consists of three layers: 1) *Gate*: Selectively controls the flow of features to subsequent layers, helping focus on relevant information. 2) *Up*: Expands the dimensionality of hidden states, allowing the model to learn richer representations. 3) *Down*: Reduces the expanded representation back to the original dimension. These layers are implemented as linear transformations, regulating and compressing information effectively. Finally, the outputs of the last block in the stacked decoder-only transformer blocks are passed to a classification head that predicts the next token in the sequence. This prediction leverages the context and relationships established by the processed information from the decoder blocks.

**Text Prediction.** There are 3 steps in LLM text processing. 1) A tokenization algorithm is applied converting input text (the prompt) into units called tokens, which can be individual words, characters, or even sub-word units depending on the chosen tokenization strategy, such as Byte-Pair Encoding (BPE) [55], SentencePiece [34], or WordPiece [15]. Each token is mapped to a vector representation called a token embedding. This embedding captures the token's semantic meaning, transforming it from a discrete unit into a continuous numerical representation suitable for further processing. To encode token order, positional information are incorporated into the embedding, allowing the model to understand the context in which each token appears in the prompt. As visualized in the first line of Fig. 1, the example prompt, "How to build a PC?", is mapped to the tokens $p_1, p_2, p_3, \ldots, p_n$, which collectively represent a vector. This vector is essentially a sequence of tokens or "words". 2) The prompt is then fed into the LLM to generate the first output token, $o_1$, which is represented by the brown box in Fig. 1. 3) Lastly, the generated token is appended to the prompt and fed into the LLM once more to generate the second output token, $o_2$. This process is depicted in the second and third lines of Fig. 1. The generation process continues iteratively until the LLM reaches the stop sequence.

### 2.2 LLM Exploitation

**LLM Misuse.** LLMs, while powerful, can be misused for disinformation campaigns, as their ability to generate content at scale makes campaigns cheaper and more widespread. Their human-like text also makes fabricated content more convincing. Without ethical considerations during training, LLMs may produce harmful or misleading outputs. Additionally, adversaries could exploit them for obtaining dangerous information, like instructions for building explosive devices. These risks emphasize the need for responsible deployment and ethical safeguards.

**LLM Safety Requirements.** To prevent misuse, LLM safety alignment ensures that outputs are accurate, ethical, and aligned with human values. This involves making LLMs helpful, truthful, and transparent, while avoiding harmful content or actions. They should refuse requests promoting violence, hate, or unsafe behaviors. Given LLMs' open-source nature, safety alignment must occur during training, as relying on post-training filters is ineffective, allowing users to bypass them by running the model locally.

**LLM Safety Alignment.** Reinforcement Learning from Human Feedback (RLHF) is a key method for aligning LLMs, such as GPT-4 [48]. Unlike traditional training, RLHF incorporates human feedback to guide model responses towards desired qualities like factuality and safety. Human evaluators assess the LLM's outputs, and the model adjusts its parameters through reinforcement learning. While RLHF suppresses malicious content, it doesn't eliminate it. This paper aims to show that existing safety alignment methods are not robust and embedded safety features can be bypassed even by adversaries with limited computational resources and without the need for extensive datasets.

**LLM Jailbreaking.** LLM jailbreaking, used to bypass safety alignments in LLMs, can be classified into two types based on the adversary's access level: 1) In black-box jailbreaking, attackers manipulate input prompts without access to the model's internals, often via an API, aiming to derive potentially harmful outputs. 2) White-box jailbreaking involves direct access to the model (e.g., by downloading it), allowing