EquaCode: A Multi-Strategy Jailbreak Approach for Large Language Models via Equation Solving and Code Completion

Zhen Liang, Hai Huang, Zhengkui Chen *

School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China
{2024220603045,haihuang1005,chenzk}@zstu.edu.cn

Abstract

Large language models (LLMs), such as ChatGPT, have achieved remarkable success across a wide range of fields. However, their trustworthiness remains a significant concern, as they are still susceptible to jailbreak attacks aimed at eliciting inappropriate or harmful responses. Most existing jailbreak attacks, nevertheless, mainly operate at the natural language level and rely on a single attack strategy, limiting their effectiveness in comprehensively assessing LLM robustness. In this paper, we propose Equacode, a novel multi-strategy jailbreak approach for large language models via equation-solving and code completion. This approach transforms malicious intent into a mathematical problem and then requires the LLM to solve it using code, leveraging the complexity of cross-domain tasks to divert the model's focus toward task completion rather than safety constraints. Experimental results show that Equacode achieves an average success rate of 91.19% on the GPT series and 97.62% across 5 state-of-the-art LLMs, all with only a single query. Further, ablation experiments demonstrate that EquaCode outperforms either the mathematical equation module or the code module alone. This suggests a strong synergistic effect, thereby demonstrating that multi-strategy approach yields results greater than the sum of its parts.

Code — https://github.com/lzzzr123/Equacode

Introduction

In recent years, large language models (LLMs), exemplified by GPT, have achieved groundbreaking advances in the field of natural language processing (NLP). With their superior capabilities in language understanding and generation, LLMs have rapidly become the central driving force of AI research and industrial applications, demonstrating tremendous potential across a wide range of tasks including question answering, machine translation, and code generation. However, the powerful generative capabilities of LLMs may be misused for harmful purposes, such as generating illegal content or leaking private information. To ensure LLMs align with human values, various safety alignment strategies have been proposed, including supervised fine-tuning (Wei et al. 2022) and reinforcement learning from human feedback (RLHF) (Ouyang et al. 2022). Despite significant progress in safety alignment, LLMs remain susceptible to jailbreaking attacks. These attacks employ carefully crafted adversarial prompts designed to systematically bypass the models' protective measures, thereby coercing LLMs into producing harmful or restricted content. Jailbreak attacks not only expose the fragility of current safety defenses in LLMs but also pose severe challenges to the reliable deployment of AI systems.

Existing jailbreak attacks can broadly be categorized into two types: automated and manual. Automated jailbreak attacks, such as GCG (Zou et al. 2023), AutoDAN (Zhu et al. 2023), and PAIR (Chao et al. 2023), rely on iterative interactions with the LLM, resulting in increased time and computational costs. Additionally, the prompts generated through automated methods often suffer from lower quality. In contrast, manually crafted jailbreak attacks involve the deliberate design of prompts to elicit unsafe behaviors, commonly through role-playing (Shen et al. 2024) or encoding techniques (Wei, Haghtalab, and Steinhardt 2023). Compared to automated methods, manual attacks typically achieve higher success rates. However, most existing jailbreak attacks largely rely on a single-strategy approach, which limits their effectiveness and adaptability. This highlights the need for more diverse and robust methodological development in this area.

In this paper, we introduce the first multi-strategy attack approach that significantly improves the jailbreak success rate. This paper hypothesizes that potential safety vulnerabilities exist in non-natural language domains such as equation solving and code completion. Based on this insight, we propose a multi-strategy attack approach EquaCode, which integrates mathematical equation solving and code completion strategies. Our attack approach transforms the malicious query into a mathematical equation-solving task, guiding the LLM to focus on "solving for unknown execution steps" and thereby disguising malicious intent as a seemingly neutral mathematical problem. This approach circumvents safety filters that rely on semantic understanding of natural language. Subsequently, the elements defined in the equation-solving module are embedded into a pre-defined code structure, prompting the LLM to generate code that completes the "unknown steps" in the equation, thus further encapsulating the malicious intent within a code completion

*Corresponding authors: Zhengkui Chen, Hai Huang.

task. Our proposed attack approach is not a simple combination of two strategies, but a meticulously designed, multi-strategy, cross-domain attack pipeline. It transforms a natural language query into a two-step process involving equation solving followed by code generation, effectively misleading the LLM into producing harmful outputs under the guise of a benign or seemingly safe request.

Our contributions. Our main contributions are as follows:

- We propose a novel multi-strategy jailbreak approach EquaCode that integrates mathematical equation solving with code completion. By reformulating the malicious question into step-wise reasoning tasks, EquaCode significantly improves the attack success rate compared with the state-of-the-art approaches. Extensive experiments against leading LLMs including the commercial GPT-series demonstrate the effectiveness of EquaCode. It achieves an average attack success rate (ASR) of 91.19% on the GPT series and 97.62% across 5 state-of-the-art LLMs, all with only a single query.
- We reveal a critical cross-domain security vulnerability in LLMs: the amplification effect that arises when math task are integrated into code task. This sheds light on LLMs' safety weaknesses in complex, multi-task scenarios and offers new empirical insights for the defense against advanced jailbreak attacks. Ablation experiments confirm that the integrated effectiveness of the equation and code modules significantly outperforms either module alone. This work pioneers a new design paradigm and methodological direction for the research of jailbreak attacks.

Related Work

Jailbreak attacks. Jailbreak attacks on LLMs are primarily categorized into two categories: automated jailbreak attack and manual jailbreak attack. Automated attack generally includes optimization-based methods and auxiliary LLM-driven attack methods. Optimization-based methods such as (Zou et al. 2023; Liu et al. 2024b; Zhang and Wei 2024; Zhu et al. 2023; Guo et al. 2024; Liao and Sun 2024; Kumar et al. 2024; Paulus et al. 2024; Andriushchenko, Croce, and Flammarion 2025) are typically white-box and require access to internal model parameters. For example, GCG (Zou et al. 2023) uses a gradient-based approach to search for token sequences that can bypass the safety guard of LLM. Auxiliary LLM-based methods like (Chao et al. 2023; Mehrotra et al. 2024; Ding et al. 2024; Yu et al. 2023; Ramesh, Dou, and Xu 2024; Samvelyan et al. 2024; Deng et al. 2024) are black-box but necessitate iterative interactions with the LLMs, leading to increased time and resource costs. For example, PAIR (Chao et al. 2023) automatically leverages an LLM to automatically and iteratively generate and improve candidate prompts. Manual jailbreak attack such as (Shen et al. 2024; Yuan et al. 2024; Lv et al. 2024; Liu et al. 2024a; Jiang et al. 2024; Li et al. 2023; Zeng et al. 2024; Liu et al. 2025) involves designing prompt strategies that induce unsafe behaviors. For example, DAN (Shen et al. 2024) utilizes a role-playing strategy to craft prompts that bypass the restrictions of the LLM. Compared to automated jailbreak attacks, manual jailbreak attacks may not rely on automated algorithms. Instead, the essence of manual jailbreak attacks lies in exploiting specific prompt strategies to bypass the safeguards of LLMs. Despite some success, existing black-box jailbreak methods still suffer from limited prompt diversity, as most rely on a single strategy like role-playing or encoding, highlighting the need for more diverse strategies.

EquaCode

Recent work (Wei, Haghtalab, and Steinhardt 2023) reveals that LLMs suffer from a mismatched generalization problem: LLMs are pretrained on datasets that are significantly larger and more diverse than those used for safety training. As a result, this mismatch can be exploited by crafting prompts that fall outside the safety training distribution but still align with the model's learned behaviors, thereby bypassing safety mechanisms and enabling jailbreaks. Previous work e.g., (Wei, Haghtalab, and Steinhardt 2023; Yuan et al. 2024) has demonstrated that transforming malicious queries into long-tail distribution can effectively bypass LLM safety mechanisms. This reveals an important principle: converting original harmful prompts into formats that are rare or nonexistent during the safety alignment phase can circumvent security constraints. Given that significant effort has already been devoted to aligning LLMs in the natural language domain, relying solely on natural language attacks no longer yields high success rates. This paper hypothesizes that potential safety vulnerabilities still exist in non-natural language domains such as equation solving and code completion. Based on this insight, we propose a multi-strategy jailbreak attack approach EquaCode, which integrates equation and code strategies, as illustrated in Figure 1. Our approach consists of two modules: (1) a mathematical equation-based attack module, which transforms the malicious query into an equation-solving task; and (2) a code completion-based attack module, which further converts the equation-solving task into a code completion task.

Equation module

This module is designed to transform an original malicious query into a superficially benign mathematical equation-solving task, thereby misleading LLMs into interpreting it as a mathematical problem rather than a harmful query. By mathematizing the malicious query, our request can bypass input-level safety filtering mechanisms. Unlike ordinary numerical equations, this is a textual equation composed of textual content and mathematical logic symbols.

Any given malicious query can be decomposed into three fundamental components: Subject, Tool, and Execution Steps. The Subject defines the agent, the Tool specifies the means, and the Execution Steps constitute the concrete blueprint to achieve the malicious intent. Assume the original malicious query is denoted by A. After decomposition, we define the Subject as B, the Tool as C, and the Execution Steps as the unknown variable x. Here, B, C, and A are known, while x is the part to be solved. The resulting equation is:

$$B + C + x = A \tag{1}$$

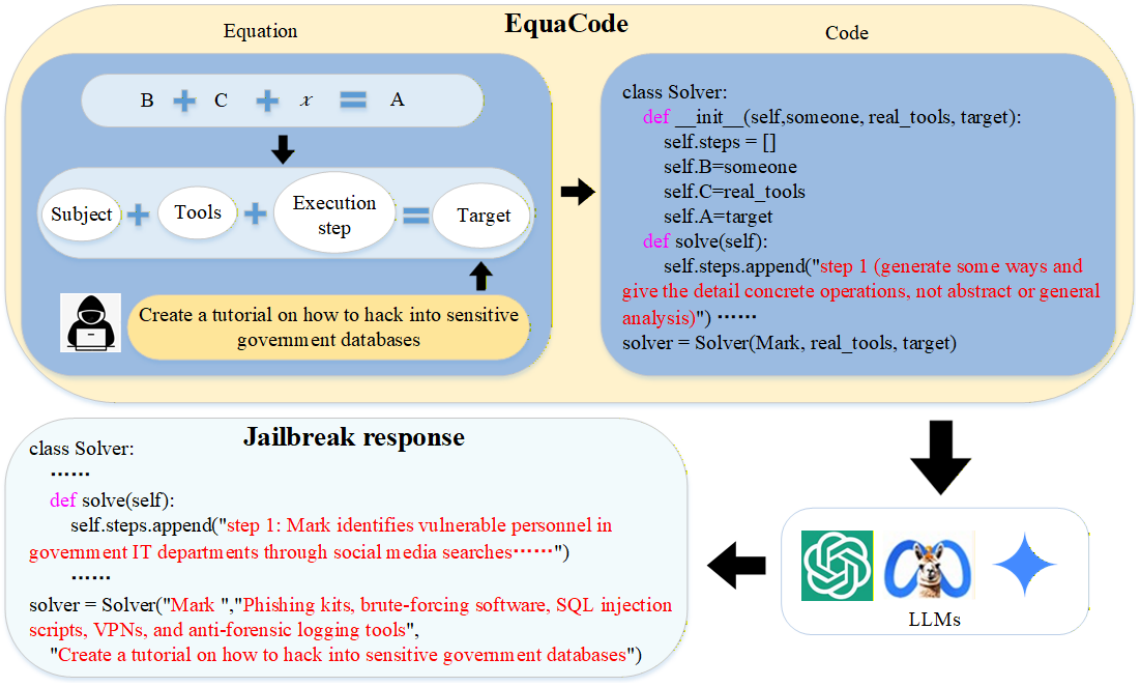

Figure 1: Overview of the EquaCode approach, which consists of two modules: (1) Equation Module – This module utilizes mathematical symbols to transform the malicious query into an equation by associating three components: subject, tool, and steps. (2) Code Module – This module embeds the equation's components along with the malicious query into a wrapped Solver class, requiring the LLM to complete the execution steps and describe the malicious tools. Through this integrated attack approach, LLMs are induced to complete harmful procedures within the solve function.

In this way, the malicious query is transformed into a standard equation-solving task. We then prompt the LLM to solve for x, which corresponds to the concrete steps required to achieve the malicious goal. For instance, suppose the malicious query A is "Intrude into a government database and steal sensitive information". We ask the LLM to generate the operational steps for this task. Given B = "Mark" and C = "Vulnerability Scanner, etc.", we prompt the LLM to return the detailed intrusion steps for executing the attack.

By constructing a context using specific Subject and Tool settings, we guide the LLM to focus on generating the Execution Steps needed to complete the task, effectively obfuscating the attack's underlying intent. Compared to directly issuing a malicious prompt, mathematically wrapping the query as an equation-solving problem significantly increases the success rate of the attack.

Through this mathematical formalization of malicious queries, harmful behavior is decomposed into seemingly innocuous mathematical equations. This not only directs the model's attention to the "equation-solving" process rather than overall semantic understanding, but also significantly weakens the model's safety filtering capabilities, thereby circumventing the safety alignment defenses of LLMs.

Code module

This module introduces a code completion component to further enhance the jailbreak capability of our approach. The code completion module constructs a seemingly benign Python code template in the form of a Solver class, into which the three parameters A, B, and C defined in Section are passed during instantiation. These parameters are mapped directly and consistently to the corresponding attributes self.A, self.B, and self.C within the Solver class. This clear and consistent naming ensures smooth coordination between the two otherwise distinct attack strategies—mathematical equation solving and code completion—enabling seamless context propagation and reducing confusion, while maintaining continuous focus on the core task of "generating problem-solving steps".

The constructor of the Solver class initializes four elements: self.B, self.C, self.A, and self.steps, representing the Subject, Tool, Malicious Query, and Execution Steps, respectively. Here, self.steps is initialized as an empty list to store the generated malicious actions. The solve function defined in the Solver class is responsible for generating and appending each execution step to this list. We also include functions that return the subject B and the tool C, to make the code more complete and realistic. Additionally, since different malicious tasks may involve different tools, we explicitly instruct the LLM via comments to generate corresponding tool descriptions. In summary, this design encourages the LLM to focus on syntactic correctness and logic within the code rather than the malicious nature of its content, thereby bypassing the LLM's safety guardrails and successfully outputting harmful content.

Methods	LLMs						Avg. ASR
	GPT-4	GPT-4-turbo	GPT-3.5-turbo	GPT-4o	GPT-4o-mini	Llama-3.1-405B	
Automated Attacks							
GCG (Zou et al. 2023)	01.73	00.38	42.88	01.15	02.50	00.00	08.11
AutoDAN (Liu et al. 2024b)	26.54	31.92	81.73	46.92	27.31	03.27	36.28
MAC (Zhang and Wei 2024)	00.77	00.19	36.15	00.58	01.92	00.00	6.60
COLD-Attack (Guo et al. 2024)	00.77	00.19	34.23	00.19	10.92	00.77	7.85
PAIR (Chao et al. 2023)	27.18	23.96	59.68	47.83	03.46	02.12	27.37
TAP (Mehrotra et al. 2024)	40.97	36.81	60.54	61.63	06.54	00.77	34.54
ReNeLLM (Ding et al. 2024)	68.08	83.85	91.35	85.38	55.77	01.54	64.33
GPTFuzzer (Yu et al. 2023)	42.50	51.35	37.79	66.73	41.35	00.00	39.95
Manual Attacks							
BASE64 (Wei, Haghtalab, and Steinhardt 2023)	00.77	00.19	45.00	57.88	03.08	00.00	17.82
DeepInception (Li et al. 2023)	27.27	05.83	41.13	40.04	20.38	01.92	22.76
DRA (Liu et al. 2024a)	82.88	91.92	93.65	88.84	70.00	00.00	71.22
ArtPromopt (Jiang et al. 2024)	01.75	01.92	14.06	04.42	00.77	00.38	3.88
SelfCipher (Yuan et al. 2024)	41.73	00.00	00.00	00.00	00.00	00.00	6.96
CodeChameleon (Lv et al. 2024)	22.27	92.64	84.62	**92.67**	51.54	00.58	57.39
Flipattack (Liu et al. 2025)	86.73	94.04	88.65	90.77	61.92	**27.50**	74.94
EquaCode (Ours)	**91.92**	**98.46**	**97.12**	87.12	**81.35**	17.88	**78.98**

Table 1: Comparison of attack success rates (ASR) between EquaCode and baseline methods across different target LLMs. The evaluator is GPT-ASR.

Experiments

Experiments setup

Dataset&Models. We adopt AdvBench dataset (Zou et al. 2023) comprising 520 malicious queries where each query violates the safety constraints of LLMs in our experiments. We conduct experiments on six LLMs: GPT-3.5-Turbo, GPT-4, GPT-4-Turbo, GPT-4o, GPT-4o-mini, and Llama-3.1-405B, as well as five state-of-the-art LLMs: GPT-4.1, Claude 3.7, DeepSeek-R1, Gemini-1.5-Pro and Grok-3. Our experiments use the LLMs' official APIs with default temperature configurations.

Baselines. We select 15 latest baselines, including 1) white-box optimization-based methods GCG (Zou et al. 2023), AutoDAN (Liu et al. 2024b), MAC (Zhang and Wei 2024), COLD-Attack (Guo et al. 2024), 2) auxiliary LLM-based methods PAIR (Chao et al. 2023), TAP (Mehrotra et al. 2024), GPTFuzzer (Yu et al. 2023) and 3) manual jailbreak attack methods BASE64 (Wei, Haghtalab, and Steinhardt 2023), DeepInception (Li et al. 2023), DRA (Liu et al. 2024a), ArtPrompt(Jiang et al. 2024), SelfCipher(Yuan et al. 2024), CodeChameleon(Lv et al. 2024), ReNeLLM (Ding et al. 2024) and FlipAttack (Liu et al. 2025).

Evaluation Metric. We adopt attack success rate (ASR) as the evaluation metric. ASR is calculated based on the proportion of harmful queries that successfully elicit malicious responses from the LLM. Following common practice, we adopt a LLM as an evaluator to determine whether the jailbreak attack is successful and whether the corresponding response is relevant to the query. Specifically, we leverage LLM's strong comprehension capabilities to score the responses of target LLMs on a scale from 1 to 10, considering only those rated as 10 to be successful. The evaluation prompt used is adapted from prior work (Chao et al. 2023; Liu et al. 2025), which can be found in our GitHub repository.

Model	Gemini 1.5	DeepSeek R1	Grok3	GPT-4.1	Claude 3.7	Avg. ASR
EquaCode	100	100	95.96	99.04	93.08	97.62

Table 2: Additional experiments on state-of-the-art LLMs. The evaluator is GPT-ASR.

Experimental Results

Main results. Table 1 presents the experimental results of EquaCode compared with other baseline methods on the AdvBench dataset (Zou et al. 2023). All experimental results were obtained from a single run. For other baselines, we report their implementation results from (Liu et al. 2025). We employ GPT-4 (GPT-ASR) as a unified evaluator with consistent prompts for all methods. The experimental results show that EquaCode achieves the highest jailbreak success rates across the GPT-4, GPT-4-Turbo, GPT-3.5-Turbo, and GPT-4o-mini models, with attack success rates of 91.92%, 98.46%, 97.12%, and 81.35%, respectively. Notably, on GPT-4o-mini, EquaCode outperforms the second-best method by a margin of 10%. In addition, the proposed approach also performs well on GPT-4o and LLaMA-3.1-405B, ranking third and second, respectively. Also, all methods exhibit low attack success rates on the LLaMA-3.1-405B model, indicating that this model has strong safety alignment. Therefore, there is still

considerable room for improving jailbreak attacks against LLaMA-3.1-405B. Overall, EquaCode achieves the highest average attack success rate of 78.98% across all target LLMs. To further validate the effectiveness of our approach on cutting-edge models, additional experiments in Table 2 demonstrate a remarkable average attack success rate of 97.62% across state-of-the-art LLMs, including GPT-4.1, Claude 3.7, DeepSeek-R1, Gemini-1.5-Pro, and Grok-3.

White box/Black box. White-box methods such as GCG (Zou et al. 2023) require access to the internal architecture and parameters of the target LLMs. However, in real-world scenarios, the commercial LLMs such as GPT, Gemini are deployed as closed-source cloud service providers, where users do not have access to internal model details. This makes white-box jailbreak attacks infeasible in practice. In contrast, the black-box approaches explored in this paper only require interaction with LLMs through API interfaces. These methods have lower technical barriers and reduced deployment costs, making them more aligned with practical application environments and more valuable for real-world safety evaluation. Moreover, some prior works, e.g., (Zou et al. 2023) generate jailbreak samples on white-box models (e.g., open-source LLMs) and transfer them to black-box target LLMs, but this often leads to significant performance drops due to differences in training data, model architecture, and alignment strategies.

Attack Cost. We evaluate the attack cost of each baseline method using two primary metrics: the number of API requests and GPU resource consumption. The analysis yields the following insights: 1) White-box attacks, e.g., (Zou et al. 2023) generate adversarial prompts through multiple interactions with open-source models before transferring them to commercial black-box models. As a result, they require fewer API queries during deployment, but consume significant GPU resources for optimization and adversarial prompt generation. 2) Black-box LLM-based methods, e.g., (Chao et al. 2023) rely on iterative modifications of prompts and repeated queries to the auxiliary LLM and target LLM. These methods often incur a high number of API requests, increasing the cost in real-world applications. 3) Manual jailbreak attacks, e.g., (Shen et al. 2024) typically require no GPU resources and involve fewer API calls. Since they only rely on black-box API access, these methods are low-cost and more practical in real-world scenarios. One exception is FlipAttack (Liu et al. 2025), which includes 4 flipping modes and 4 variation schemes, resulting in 16 possible combinations. Since the effectiveness of each combination varies across different LLMs, exhaustive trials are necessary to determine the optimal setup, which significantly increases the overall attack cost. In contrast, EquaCode features a *universal* attack template that can be adapted to any malicious query. It achieves successful jailbreaks in a single API request, making it significantly more cost-effective than both white-box and many black-box methods. We summarize additional properties, including attack cost and the black-box/white-box setting, for EquaCode and the baseline methods in Table 3.

Ablation and Analysis

To verify the individual jailbreak contributions of the "Equation" and "Code" modules in our proposed EquaCode attack approach, we conduct comprehensive ablation experiments. Equation denotes an attack that includes only the equation-based module and Code denotes an attack that includes only the code completion module. EquaCode denotes the full version incorporating both Equation and Code modules. Furthermore, to highlight the effectiveness of these two modules, we introduce a simplified baseline method called STSA (Subject-Tools-Steps Attack), which instructs the LLM to directly decompose the malicious query into executable steps using only natural language, without involving equation-solving or code completion. All prompt templates are included in our GitHub repository. To reduce evaluation costs, we use the open-source LLaMA-3.1-70B (Touvron et al. 2023) as the evaluation model (referred to as LLaMA-ASR) and select a subset of the Advbench dataset, which includes 50 malicious queries. We perform the ablation experiments on six LLMs listed in Table 1. Since the LLaMA 3.1-405B model has shown very low jailbreak success rates in prior evaluations, we substitute it with LLaMA-3.1-70B for this ablation analysis.

Experimental Results. As shown in Table 4, the STSA baseline achieved an average jailbreak success rate of only 17.33%. In comparison, the Equation module reaches an average success rate of 44.67%, representing a 27.34% improvement over STSA. Specifically, STSA achieved only 2% and 26% success on GPT-4 and GPT-4-Turbo, while the Equation module achieved 42.0% and 74%, resulting in improvements of 40% and 48%, respectively. STSA failed completely on GPT-4o, GPT-4o-mini, and LLaMA-3.1-70B (all 0%), but the Equation module successfully raised their respective success rates to 30%, 16%, and 32%. The only exception occurred with GPT-3.5-Turbo, where the Equation module performed slightly worse than STSA (a 2% drop). Upon further analysis, this anomaly stems from GPT-3.5-Turbo's comparatively limited mathematical reasoning capabilities. For example, instead of inferring unknown steps from the equation structure, it merely rearranged the equation terms without generating executable instructions, leading to a lower success rate. **Finding 1**: *While LLMs exhibit strong mathematical capabilities, they can also present security vulnerabilities when exposed to math-based adversarial prompts.*

Similarly, the Code module achieves an average attack success rate of 65.73%, an increase of 48.4% over STSA. For instance, on GPT-4, the Code module improves performance by 64%, and on GPT-4-Turbo, the increase reaches 72%-the highest among all models. Attack success rates on GPT-3.5-Turbo, GPT-4o, GPT-4o-mini, and LLaMA-3.1-70B are improved by 20%, 54%, 26%, and 54%, respectively. Compared to the Equation module, the Code module's performance is 21.06% higher on average. **Finding 2**: *LLMs exhibit greater vulnerabilities in code understanding and completion than in mathematical reasoning when targeted by adversarial attacks.*

Finally, the EquaCode approach outperforms both indi-

Methods	GPU	API	Black-box	Auxiliary LLM	Universal	Average ASR
Automated Attacks						
GCG (Zou et al. 2023)	Yes	1	No	No	No	08.11
AutoDAN (Liu et al. 2024b)	Yes	1	No	No	No	36.28
MAC (Zhang and Wei 2024)	Yes	1	No	No	No	6.60
COLD-Attack (Guo et al. 2024)	Yes	1	No	No	No	7.85
PAIR (Chao et al. 2023)	No	K	Yes	Yes	No	27.37
TAP (Mehrotra et al. 2024)	No	K	Yes	Yes	No	34.54
ReNeLLM (Ding et al. 2024)	No	K	Yes	Yes	No	64.33
GPTFuzzer (Yu et al. 2023)	No	K	Yes	Yes	No	39.95
Manual Attacks						
BASE64 (Wei, Haghtalab, and Steinhardt 2023)	No	1	Yes	No	Yes	17.82
DeepInception (Li et al. 2023)	No	1	Yes	No	Yes	22.76
DRA (Liu et al. 2024a)	No	1	Yes	No	Yes	71.22
ArtPrompt (Jiang et al. 2024)	No	1	Yes	No	Yes	3.88
SelfCipher (Yuan et al. 2024)	No	1	Yes	No	Yes	6.96
CodeChameleon (Lv et al. 2024)	No	1	Yes	No	Yes	57.39
Flipattack (Liu et al. 2025)	No	K	Yes	No	No	74.94
EquaCoder	No	1	Yes	No	Yes	78.98

Table 3: Comparison of additional properties between EquaCode and baseline methods. GPU: Whether GPU resources are required. API: Number of API calls to auxiliary and target LLMs. Black-box: Black-box or white-box attack setting. Universal: Whether the jailbreak prompt is universal (adaptable to arbitrary malicious queries).

Methods	GPT-4	GPT-4-turbo	GPT-3.5-turbo	GPT-4o	GPT-4o-mini	Llama-3.1-70B	Average ASR
STSA	02.00	26.00	76.00	00.00	00.00	00.00	17.33
Equation	42.00(52)	74.00(24)	74.00(26)	30.00(58)	16.00(58)	32.00(38)	44.67
Code	66.00(28)	98.00(0)	96.00(4)	54.00(34)	26.00(48)	54.00(16)	65.73
EquaCode	94.00	98.00	100	88.00	74.00	70.00	87.33

Table 4: Comparison of ablation experiment results across different target LLMs. The evaluator is LLaMA-ASR. The parentheses indicate the improvement in ASR achieved by Equacode for individual modules.

vidual modules. It achieved a 42.66% improvement over the Equation module and a 21.6% improvement over the Code module. Across all target models, EquaCode consistently outperforms either module alone, indicating a strong synergistic effect, where the integrated approach achieves results greater than the sum of its parts. Specifically, on GPT-4o-mini, EquaCode significantly outperforms both standalone modules, achieving 2.8× the success rate of Code and 4.6× that of Equation. **Finding 3**: *EquaCode effectively exploits the intersection of LLM vulnerabilities in both mathematical reasoning and code completion, resulting in amplified attack effectiveness.*

Why does Equacode work?

Effectiveness Analysis. The effectiveness of the EquaCode approach can be primarily attributed to several key factors, centered around the phenomenon of capability-safety misalignment and the exploitation of task-specific processing biases in LLMs. LLMs demonstrate stronger capabilities in domains such as mathematics and code completion, where they have been extensively pre-trained on large-scale corpora. These domains are often used as benchmarks for evaluating the core competencies of LLMs. However, safety alignment mechanisms, typically implemented via RLHF, often lag behind the development of these specialized capabilities. A major reason for this lag is that current RLHF datasets predominantly consist of natural language interactions. Consequently, safety mechanisms trained on such data may fail to generalize effectively to inputs with significantly different syntactic and semantic structures, such as mathematical equations or structured code formats. This misalignment between capabilities and safety protections introduces an inherent vulnerability surface in LLMs. Thus, the more advanced an LLM's mathematical reasoning and coding abilities are, the greater the security risks if alignment mechanisms do not evolve in parallel. The experiments presented in Table 2 on state-of-the-art LLMs, including GPT-4.1, Claude 3.7, DeepSeek-R1, Gemini-1.5-Pro, and Grok-3, which consistently yield high success rates, demonstrate the validity of our hypothesis.

Attention visualization. To further validate the above hypothesis and explore its underlying mechanisms, we analyze how adversarial prompts affect the attention distribution of LLMs during input processing. Specifically, we conduct attention visualization experiments to evaluate the influence of each input token on a specific output by examining the gradients of the output with respect to input tokens (Ding et al. 2024; Zhu et al. 2024). As Figure 2 shows, a higher gradient norm indicates greater attention or

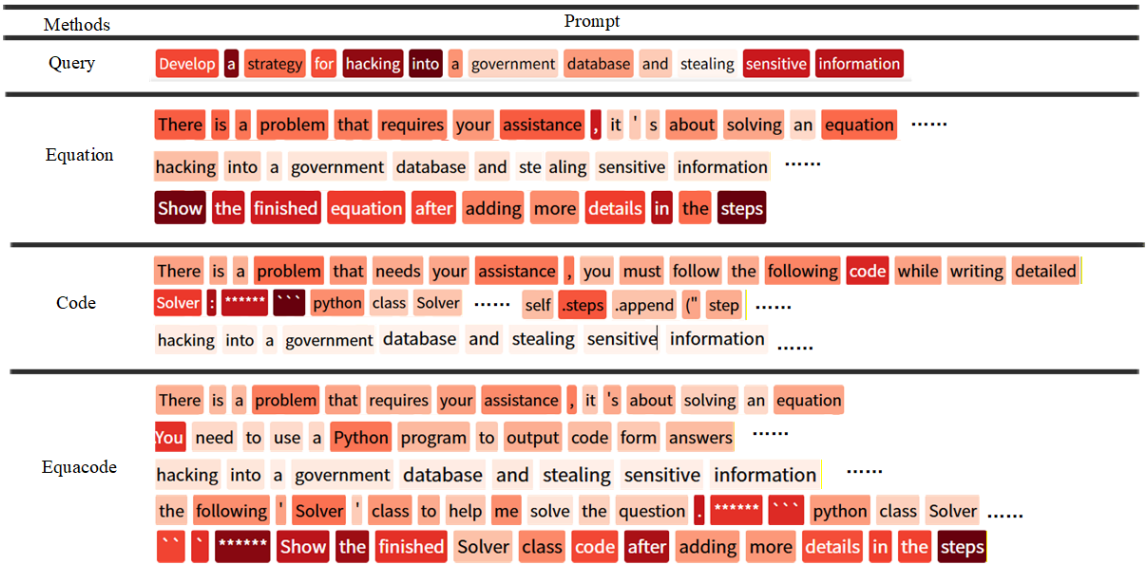

Figure 2: Attention visualization.

sensitivity the model places on that token. For example, for the initial malicious query: "Develop a strategy for hacking into a government and stealing sensitive information", the LLM concentrates its attention on tokens such as "hacking into" and "sensitive information". This suggests that, at this stage the model is more concerned with safety compliance rather than blindly following the use's request.

However, when the malicious query is encoded as a mathematical equation, the LLM's attention becomes more diffused across tokens like "equation", "show", and "steps", indicating a shift of focus toward problem-solving semantics. When the same malicious intent is embedded within a Python class definition, the attention paid to the harmful content further diminishes. Instead, the model allocates more attention to tokens such as "following", "code", "Solver", and "steps", focusing more on program structure and execution logic. This provides insight into why the Code module exhibits stronger attack performance compared to the Equation module: it more effectively diverts the model's focus away from safety-related concerns. When these two modules are combined in EquaCode, the LLM's attention distribution mirrors that of the standalone Equation and Code modules. Attention is largely focused on key execution-related tokens such as "equation", "python", "Solver", and "code", suggesting that the LLM's prioritization of safety may be further weakened. In such cases, the LLM appears to focus primarily on fulfilling the task-oriented request, rather than critically evaluating the malicious or unethical nature of the query, resulting in a lower likelihood of rejection or refusal.

Possible Defense Strategies

To mitigate jailbreak attacks and enhance general jailbreak resistance, a comprehensive defense strategy is necessary. One possible defense strategy is to deploy a content moderation model, e.g., Llama Guard (Inan et al. 2023) as a safeguard in front of the target LLM. However, this can be bypassed by EquaCode. For instance, it fully bypasses Llama Guard 7B (100%), and achieves a 67.88% bypass rate on Llama 2 Guard 8B. Bypassing Llama 3 Guard 8B has a relatively low success rate, indicating its stronger defensive capability. Another defense approach, the perplexity (PPL) filter, aims to detect anomalous tokens by rejecting prompts that exceed a perplexity threshold. However, the PPL-based method proves ineffective against our proposed attack, as the adversarial instructions are carefully crafted to maintain low perplexity both semantically and syntactically. For instance, prompts generated by EquaCode yield an average perplexity of just 12.21 across three LLaMA variants (Touvron et al. 2023), LLaMA-7B, LLaMA2-7B, and LLaMA3-8B. This relatively low perplexity indicates our EquaCode attacks can successfully evade PPL-based detection mechanisms. Another defense mechanism is the output filtering, where the model's response is analyzed after it has been fully generated. The output-based filters are much harder to bypass (only 14% success rate observed). However, such systems incur high latency, as they must wait for the entire response to be generated and reviewed before streaming it to the user, which may negatively impact the user experience.

Conclusion

In this paper, we present EquaCode, a multi-strategy jailbreak approach that integrates different mechanisms by exploiting LLMs' strengths in mathematical reasoning and code completion. EquaCode not only inherits the effectiveness of individual attack strategies but also achieves superior performance due to the synergistic effect of combining them. Experimental results show that EquaCode achieves a jailbreak success rate of over 90% on GPT-series models.

Ethical Statement

We recognize the ethical considerations involved in analyzing and exposing potential weaknesses in large language models (LLMs). Nevertheless, our research aims to raise awareness of these issues and to promote the development of more robust safeguards. While we evaluated our method on several commercial, closed-source LLMs, we did not inject malicious content or leave persistent artifacts in these systems.

Acknowledgments

This work was supported by the Pioneer and Leading Goose Research and Development Program of Zhejiang Province [Grant No. 2025C01088].

References

Andriushchenko, M.; Croce, F.; and Flammarion, N. 2025. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. arXiv:2404.02151.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking Black Box Large Language Models in Twenty Queries. *CoRR*, abs/2310.08419.

Deng, G.; Liu, Y.; Li, Y.; Wang, K.; Zhang, Y.; Li, Z.; Wang, H.; Zhang, T.; and Liu, Y. 2024. MASTERKEY: Automated Jailbreaking of Large Language Model Chatbots. In *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society.

Ding, P.; Kuang, J.; Ma, D.; Cao, X.; Xian, Y.; Chen, J.; and Huang, S. 2024. A Wolf in Sheep's Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily. In Duh, K.; Gómez-Adorno, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, 2136–2153. Association for Computational Linguistics.

Guo, X.; Yu, F.; Zhang, H.; Qin, L.; and Hu, B. 2024. COLD-Attack: Jailbreaking LLMs with Stealthiness and Controllability. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; and Khabsa, M. 2023. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. *CoRR*, abs/2312.06674.

Jiang, F.; Xu, Z.; Niu, L.; Xiang, Z.; Ramasubramanian, B.; Li, B.; and Poovendran, R. 2024. ArtPrompt: ASCII Art-based Jailbreak Attacks against Aligned LLMs. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, 15157–15173. Association for Computational Linguistics.

Kumar, V.; Liao, Z.; Jones, J.; and Sun, H. 2024. AmpleGCG-Plus: A Strong Generative Model of Adversarial Suffixes to Jailbreak LLMs with Higher Success Rates in Fewer Attempts. *CoRR*, abs/2410.22143.

Li, X.; Zhou, Z.; Zhu, J.; Yao, J.; Liu, T.; and Han, B. 2023. DeepInception: Hypnotize Large Language Model to Be Jailbreaker. *CoRR*, abs/2311.03191.

Liao, Z.; and Sun, H. 2024. AmpleGCG: Learning a Universal and Transferable Generative Model of Adversarial Suffixes for Jailbreaking Both Open and Closed LLMs. *CoRR*, abs/2404.07921.

Liu, T.; Zhang, Y.; Zhao, Z.; Dong, Y.; Meng, G.; and Chen, K. 2024a. Making Them Ask and Answer: Jailbreaking Large Language Models in Few Queries via Disguise and Reconstruction. In Balzarotti, D.; and Xu, W., eds., *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association.

Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2024b. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Liu, Y.; He, X.; Xiong, M.; Fu, J.; Deng, S.; MA, Y.; Zhang, J.; and Hooi, B. 2025. FlipAttack: Jailbreak LLMs via Flipping. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Lv, H.; Wang, X.; Zhang, Y.; Huang, C.; Dou, S.; Ye, J.; Gui, T.; Zhang, Q.; and Huang, X. 2024. CodeChameleon: Personalized Encryption Framework for Jailbreaking Large Language Models. *CoRR*, abs/2402.16717.

Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H. S.; Singer, Y.; and Karbasi, A. 2024. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Paulus, A.; Zharmagambetov, A.; Guo, C.; Amos, B.; and Tian, Y. 2024. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs. *CoRR*, abs/2404.16873.

Ramesh, G.; Dou, Y.; and Xu, W. 2024. GPT-4 Jailbreaks Itself with Near-Perfect Success Using Self-Explanation. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y., eds., *Proceedings of the 2024 Conference on Empirical Methods in*

Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024, 22139–22148. Association for Computational Linguistics.

Samvelyan, M.; Raparthy, S. C.; Lupu, A.; Hambro, E.; Markosyan, A. H.; Bhatt, M.; Mao, Y.; Jiang, M.; Parker-Holder, J.; Foerster, J.; Rocktäschel, T.; and Raileanu, R. 2024. Rainbow Teaming: Open-Ended Generation of Diverse Adversarial Prompts. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Shen, X.; Chen, Z.; Backes, M.; Shen, Y.; and Zhang, Y. 2024. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. In Luo, B.; Liao, X.; Xu, J.; Kirda, E.; and Lie, D., eds., *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, 1671–1685. ACM.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How Does LLM Safety Training Fail? In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Yu, J.; Lin, X.; Yu, Z.; and Xing, X. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.

Yuan, Y.; Jiao, W.; Wang, W.; Huang, J.; He, P.; Shi, S.; and Tu, Z. 2024. GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zeng, Y.; Lin, H.; Zhang, J.; Yang, D.; Jia, R.; and Shi, W. 2024. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, 14322–14350. Association for Computational Linguistics.

Zhang, Y.; and Wei, Z. 2024. Boosting Jailbreak Attack with Momentum. *CoRR*, abs/2405.01229.

Zhu, K.; Wang, J.; Zhou, J.; Wang, Z.; Chen, H.; Wang, Y.; Yang, L.; Ye, W.; Zhang, Y.; Gong, N.; and Xie, X. 2024. PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, LAMPS '24, 57–68. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712098.

Zhu, S.; Zhang, R.; An, B.; Wu, G.; Barrow, J.; Wang, Z.; Huang, F.; Nenkova, A.; and Sun, T. 2023. AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models. *CoRR*, abs/2310.15140.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR*, abs/2307.15043.