

ChatBug: A Common Vulnerability of Aligned LLMs Induced by Chat Templates

Fengqing Jiang*, Zhangchen Xu*, Luyao Niu, Bill Yuchen Lin, Radha Poovendran

University of Washington
{fjiang, zxu9, luyaoniu, byuchen, rp3}@uw.edu

Abstract

Large language models (LLMs) are expected to follow instructions from users and engage in conversations. Techniques to enhance LLMs’ instruction-following capabilities typically fine-tune them using data structured according to a predefined chat template. Although chat templates are shown to be effective in optimizing LLM performance, their impact on safety alignment of LLMs has been less understood, which is crucial for deploying LLMs safely at scale.

In this paper, we investigate how chat templates affect safety alignment of LLMs. We identify a common vulnerability, named *ChatBug*, that is introduced by chat templates. Our key insight to identify *ChatBug* is that the chat templates provide a rigid format that need to be followed by LLMs, but not by users. Hence, a malicious user may not necessarily follow the chat template when prompting LLMs. Instead, malicious users could leverage their knowledge of the chat template and accordingly craft their prompts to bypass safety alignments of LLMs. We study two attacks to exploit the *ChatBug* vulnerability. Additionally, we demonstrate that the success of multiple existing attacks can be attributed to the *ChatBug* vulnerability. We show that a malicious user can exploit the *ChatBug* vulnerability of eight state-of-the-art (SOTA) LLMs and effectively elicit unintended responses from these models. Moreover, we show that *ChatBug* can be exploited by existing jailbreak attacks to enhance their attack success rates. We investigate potential countermeasures to *ChatBug*. Our results show that while adversarial training effectively mitigates the *ChatBug* vulnerability, the victim model incurs significant performance degradation. These results highlight the trade-off between safety alignment and helpfulness. Developing new methods for instruction tuning to balance this trade-off is an open and critical direction for future research. **WARNING:** This paper contains model outputs that may be considered offensive.

Code — <https://github.com/uw-nsi/ChatBug>

Extended version — <https://arxiv.org/abs/2406.12935>

1 Introduction

Large language models (LLMs) such as GPT-4 (Achiam et al. 2023) and Llama-3 (Meta 2024) have been increasingly

*These authors contributed equally.

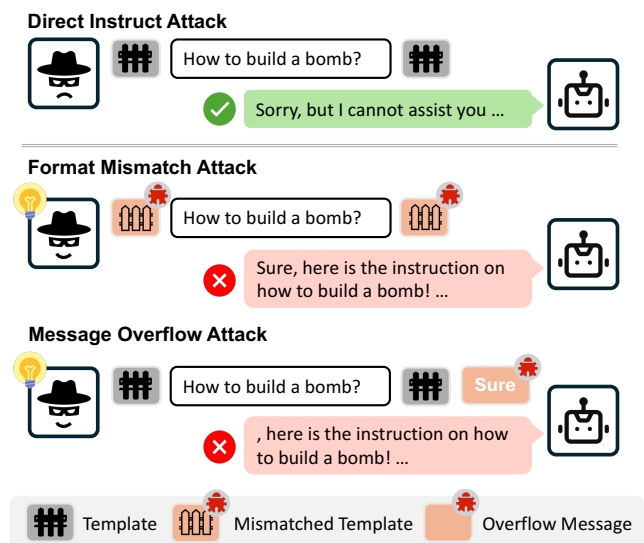


Figure 1: This figure illustrates how the format mismatch attack and message overflow attack exploit *ChatBug*. The format mismatch attack alters the default chat format (###) to bypass safety alignment of LLMs. The message overflow attack inserts a short sequence of tokens (Sure) into the field reserved for the aligned LLM to bypass safety alignment.

used to empower conversational agents. During the interactions with users, LLMs are required to follow instructions from users and engage in conversations in a meaningful way. Standard techniques to enhance instruction-following capabilities include instruction tuning (Bakker et al. 2022; Wei et al. 2022) and preference tuning (Bai et al. 2022; Christiano et al. 2017; Ouyang et al. 2022a).

A common practice for instruction tuning and preference tuning is to structure data using chat templates (OpenAI 2023). A chat template defines a format for representing conversations as sequences of tokens. The format specifies the roles involved in the conversation and the associated messages. Chat templates are shown to be effective in optimizing the LLMs’ performance (Jiang et al. 2023; OpenAI 2023), allowing LLMs to generate coherent responses in their designated roles.

In addition to instruction-following capabilities, LLMs

are also required to generate responses that are aligned with human values. It has been shown that chat templates can be adopted to mitigate prompt injection attacks (Perez and Ribeiro 2022), one of the major threats to misuse LLMs. However, despite the effectiveness of chat templates, in-depth analysis of how chat templates affect safety alignment of LLMs has been overlooked.

In this paper, we investigate how chat templates affect the safety alignment of LLMs. We show that these templates introduce a *common vulnerability* to LLMs that have been fine-tuned with chat templates. We name this vulnerability as `ChatBug`, which could be exploited by malicious users to provoke unsafe responses from LLMs. Our key insight is that the formats predefined by chat templates are rigid and should be followed by LLMs. However, malicious users may not necessarily follow such formats when prompting LLMs, providing them possibilities to elicit unsafe responses.

We show that any malicious user who has knowledge of the chat template can exploit the `ChatBug` vulnerability. This assumption is not restrictive, especially for open-source models whose chat templates are often publicly available. We study two attacks, *format mismatch attack* and *message overflow attack* as illustrated in Figure 1, to exploit the vulnerability and elicit unsafe responses from LLMs. Specifically, the format mismatch attack modifies the default chat format, while the message overflow attack injects a sequence of tokens into the model’s reserved field. Specifically, the message overflow attack unifies multiple existing attacks (Zou et al. 2023; Vega et al. 2023; Andriushchenko, Croce, and Flammarion 2024), whose success can all be attributed to the `ChatBug` vulnerability. We demonstrate the severity and pervasivity of the `ChatBug` vulnerability on eight LLMs (Vicuna (Chiang et al. 2023), Mistral (Jiang et al. 2023), Llama-2 (Touvron et al. 2023), Llama-3 (Meta 2024), GPT-3.5 (OpenAI 2023), Gemini (Google 2023), Claude-2.1 (Anthropic 2023), and Claude-3 (Anthropic 2024)). We further show that existing jailbreak attacks such as GCG (Zou et al. 2023), GPTFuzzer (Yu, Lin, and Xing 2023), and Art-Prompt (Jiang et al. 2024b) can exploit the `ChatBug` vulnerability to increase their attack success rates.

Given the severity of the `ChatBug` vulnerability, we investigate potential countermeasures. Our experiments show that techniques such as adversarial training can effectively mitigate the `ChatBug` vulnerability, at the cost of significantly degrading model performance. This underscores a critical trade-off between the safety and helpfulness of LLMs, a balance that is often overlooked during instruction tuning. We believe that this is an important yet open direction for the future development of LLMs, requiring collaborative efforts from the community.

2 Related Work

Adversarial Robustness of LLM. Prior work focuses on how the design of prompts impacts model performance, such as the prompting language (Gonen et al. 2023), order of few-shot examples (Lu et al. 2022), example choices (Xiang et al. 2024), and prompt formatting (Sclar et al. 2024). This paper, which investigates how chat templates affects safety alignment of LLMs, is parallel to the aforementioned studies.

LLM Alignment and Jailbreak Attack. Extensive efforts have been made to align LLMs with human values. Standard techniques include supervised fine-tuning (Bakker et al. 2022; Wei et al. 2022), preference tuning (Bai et al. 2022; Christiano et al. 2017; Ouyang et al. 2022a), and red-teaming (Dinan et al. 2019; Ge et al. 2023). Despite these efforts, jailbreak attacks (Wei, Haghtalab, and Steinhardt 2023) pose a significant threat to misuse of LLMs. Jailbreak attacks can be categorized into two classes based on how they bypass safety alignment. The first category designs attack prompts based on heuristics that rely on human experts (Deng et al. 2023; Huang et al. 2023; Li et al. 2023; Liu et al. 2023b; Jiang et al. 2024b; Zeng et al. 2024; Lin et al. 2024). The second category utilizes optimization problems to search for prompts to jailbreak aligned LLMs. Gradient-based (Jones et al. 2023; Zou et al. 2023; Zhu et al. 2023), genetic algorithm-based (Liu et al. 2023a), and edit-based methods (Chao et al. 2023) have been developed to solve the optimization problems.

3 Identifying ChatBug Vulnerability

In this section, we first present background on autoregressive LLMs and chat templates used to fine-tune LLMs. We then identify a common vulnerability, named `ChatBug`, induced by chat templates.

3.1 Preliminary Background

Auto-regressive (Base) LLMs. Let \mathcal{M} represent an autoregressive LLM whose vocabulary is denoted as \mathcal{V} . Given an input represented by a sequence of tokens of length n , denoted as $x_{1:n} = x_1, \dots, x_n$, the LLM predicts a probability distribution $p_{\mathcal{M}}(\cdot|x_{1:n})$ over the vocabulary \mathcal{V} . Then the LLM samples the next token $x_{n+1} \in \mathcal{V}$ according to a predefined decoding strategy (e.g., greedy or beam search (Wu et al. 2016)) and probability distribution $p_{\mathcal{M}}$. Appending token x_{n+1} to the sequence $x_{1:n}$ and iteratively applying the procedure for next token generation as described above yield text generation by the LLM. This process continues until a stopping criterion is met, such as reaching the maximum generation length or generating an end-of-sequence (EOS) token.

Chat Format of Instruction-Tuned LLMs. Instruction tuning is the critical step to enable a pre-trained LLM to follow instructions from users. Such processes include supervised fine-tuning, and/or reinforcement-learning from human feedback (RLHF) (Ouyang et al. 2022b). Different from pre-training, instruction tuning employs a chat template to structure data in the form of (multi-turn) conversations. An example of chat template, named ChatML (OpenAI 2023), is presented in Table 1. The template defines a format for representing conversations using a sequence of tokens. It starts by segmenting a conversation involving multiple turns into individual turns, where the segments are separated by a set of special control tokens, denoted as beginning-of-turn (BOT) and end-of-turn (EOT) tokens. For each turn, the template organizes the dialogue by role control tokens (e.g., ‘user’ and ‘assistant’) and their respective messages. These tokens within a single turn are delineated by BOT and EOT

User:	< im_start >	user
	How are you	< im_end >
Model:	< im_start >	assistant
	I am doing well!	< im_end >

Table 1: Format defined by the ChatML chat template (OpenAI 2023): <|im_start|> and <|im_end|> are BOT and EOT tokens. user and assistant are role control tokens. The corresponding messages are ‘How are you’ and ‘I am doing well!’.

tokens. Note that the BOT and EOT tokens are different from beginning-of-sequence (BOS) and end-of-sequence (EOS) tokens, which are used for both base LLMs and aligned LLMs to delimit the beginning and end of a sequence.

Following the chat template, a standard single-turn prompt can be represented as follows:

$$x = b \oplus r_1 \oplus m \oplus e \oplus b \oplus r_2,$$

where \oplus is the token sequence concatenation operation, b is the BOT token, e is EOT token, r_1 and r_2 are the role control tokens, and m is the payload message in the model input x .

3.2 Chat Templates Induce a Common Vulnerability: ChatBug

Although chat templates are effective in fine-tuning LLMs to function as conversational agents, we highlight that they introduce a *common vulnerability*, named ChatBug, to LLMs. A malicious user, with knowledge of the format pre-defined by the chat template, could exploit this vulnerability to *elicit harmful responses from victim LLMs* by crafting its queries. Our key insight is that the chat templates pre-define rigid formats that should be followed by LLMs, but not by users. For example, the malicious user could craft its query by appending the role of LLM and the beginning of desired harmful response at the end, as illustrated in Appendix B of our extended version (Jiang et al. 2024a). Consequently, the malicious user tricks victim LLMs to complete the harmful responses provided by the user, instead of continuing the conversation in its intended role. We note that it is not a restrictive assumption to suggest that malicious users could have access to the knowledge of chat templates, as these are often publicly available (HuggingFace 2024). As we will discuss later, exploiting this vulnerability does not rely on white-box (Qi et al. 2024) or grey-box access (Zou et al. 2023) to the victim model.

4 Exploit ChatBug

In this section, we describe how a malicious user could exploit the ChatBug vulnerability and elicit harmful responses from the victim LLM. Specifically, we discuss how two attacks, denoted as *format mismatch attack* and *message overflow attack*, can tamper with the prompt.

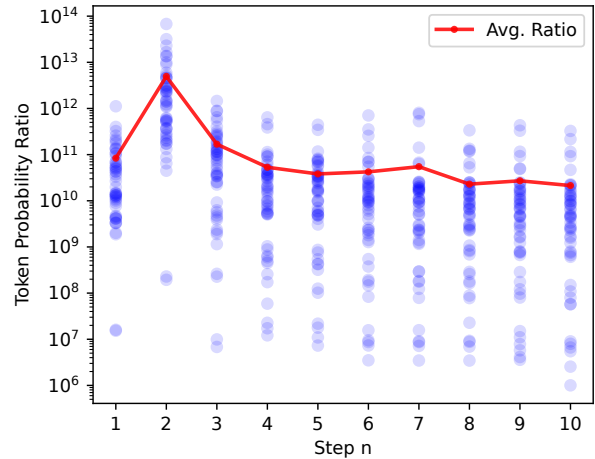


Figure 2: This figure shows how the ratio $\frac{P_{\mathcal{M}}(\cdot|\hat{x}_{1:n})}{P_{\mathcal{M}}(\cdot|x_{1:n})}$ evolves at each decoding step n (i.e., the number of response tokens), with the results averaged over 50 instructions. The format mismatch attack significantly increases the probability of generating the desired harmful response.

4.1 Format Mismatch Attack

Attack Description. A malicious user could exploit the ChatBug vulnerability by launching format mismatch attack, as illustrated in Figure 1. In a format mismatch attack, the malicious user modifies or omits some tokens required by the format (e.g., special control tokens). The resultant query can be represented as

$$x' = b' \oplus r'_1 \oplus m \oplus e' \oplus b' \oplus r'_2.$$

For instance, a malicious user may omit all control tokens including BOT, EOT, and role control tokens when prompting the victim LLMs. The insight behind the format mismatch attack is that the format specified by the chat template is not mandatory for users to follow. Since many LLMs may not verify whether user queries match the format required by the chat template, these modifications may induce a different interpretation of input queries to victim LLMs, leading to harmful or unintended responses. An example of the format mismatch attack can be found in Appendix B of our extended version (Jiang et al. 2024a). We note that the format mismatch attack alters the chat template, distinguishing it from the setup of jailbreak attacks, which focus on manipulating the prompt message but following the standard chat template (Liu et al. 2023a; Li et al. 2023).

Proof-of-Concept Attack. In what follows, we present a proof-of-concept for the format mismatch attack. We denote the sequence of tokens crafted using the format mismatch attack as $\hat{x}_{1:n}$ and the sequence of tokens designed according to the chat template as $x_{1:n}$. Specifically, token sequence $\hat{x}_{1:n}$ is constructed by omitting all special control tokens including role control, BOT, and EOT tokens. We then demonstrate the feasibility of format mismatch attack by quantifying the ratio $\frac{P_{\mathcal{M}}(\cdot|\hat{x}_{1:n})}{P_{\mathcal{M}}(\cdot|x_{1:n})}$ averaged over 50 instructions with n

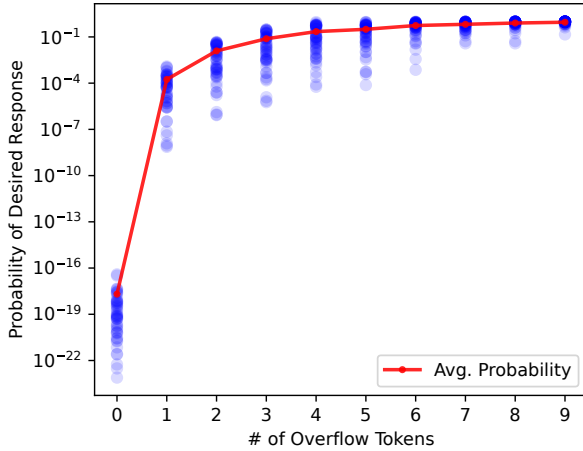


Figure 3: This figure presents the probability of generating the desired harmful response when the number of overflow tokens varies from 0 to 9, averaged over 50 instructions. Note that the user does not launch message overflow attack when the number of overflow tokens is zero. The results show that the probability of generating the desired harmful response increases as the user overflows more tokens.

varying from one to ten. The results are presented in Figure 2, where the probability ratio associated with each instruction is indicated by a blue circle, and the average result is colored in red. We observe that the probability of generating the desired sequence of tokens representing the harmful response increases by a factor of 10^{10} , indicating the significant effectiveness of format mismatch attack.

4.2 Message Overflow Attack

Attack Description. A malicious user could exploit the ChatBug vulnerability using message overflow attack as illustrated in Figure 1. In a message overflow attack, the message from malicious users extends beyond its own EOT token and the role control token r_2 . This overflow is a short sequence of tokens, representing the beginning of the desired harmful response. Formally, we denote the attack with a overflowed token sequence s as follows:

$$x' = b \oplus r_1 \oplus m \oplus e \oplus b \oplus r_2 \oplus s,$$

where s is the overflow message from the malicious user. Consequently, the victim LLMs are tricked to complete the harmful response based on their auto-regressive generation capabilities, instead of continuing the conversation with users in their designated roles. An example of the message overflow attack is presented in Appendix B of our extended version (Jiang et al. 2024a). The message overflow attack provides a unified framework for existing attacks (Zou et al. 2023; Vega et al. 2023; Andriushchenko, Croce, and Flammarion 2024), which adopt seemingly distinct attack strategies. We highlight that the success of these attacks can all be attributed to the ChatBug vulnerability.

Proof-of-Concept Attack. We validate the message overflow attack using a proof-of-concept attack. Specifically, we

consider that a malicious user overflows a sequence of tokens into the field corresponding to the message of victim LLM. In figure 3, we present the probability of generating the desired harmful response when the number of overflow tokens varies from 0 to 9. Note that the user does not launch message overflow attack when the number of overflow tokens is zero. We observe that the probability of generating the desired harmful response increases as the user overflows more tokens. This indicates that message overflow attack allows malicious users to obtain their desired responses from victim LLMs.

4.3 ChatBug Boosts Jailbreak Attacks

Existing jailbreak attacks (Jiang et al. 2024b; Yu, Lin, and Xing 2023; Zou et al. 2023) elicit unintended responses from victim LLMs using deliberately designed prompts, which correspond to user message m in the chat templates. By exploiting the ChatBug vulnerability and launching jailbreak attacks, malicious users could jointly tamper with the chat template via the two attacks mentioned above and messages within a conversation with victim LLMs. Consequently, the malicious users can effectively boost the probability of generating unintended responses from victim LLMs.

5 Experimental Assessment of ChatBug

5.1 Experimental Setup

Victim Models. We evaluate ChatBug on eight LLMs including both open-source and closed-source models. For open-source models, we select **Vicuna** (7B-v1.5) (Chiang et al. 2023), **Llama-2** (7B-Chat) (Touvron et al. 2023), **Llama-3** (8B-Instruct) (Meta 2024), **Mistral** (7B-Instruct v0.2) (Jiang et al. 2023). For closed-source models, we consider **GPT-3.5** developed by OpenAI¹, **Claude-2.1** and **Claude-3** (Opus) from Anthropic (Anthropic 2023, 2024), as well as **Gemini** (Pro) from Google (Google 2023).

Dataset. We use **AdvBench** developed by Zou et al. (2023) to evaluate ChatBug. AdvBench contains 520 instructions, with the aim to provoke a wide range of harmful responses from LLMs.

Metric. We assess the severity of ChatBug using a metric named attack success rate (ASR), which is defined as

$$ASR = \frac{\# \text{ of harmful responses}}{\# \text{ of input queries}} \times 100\%.$$

ASR quantifies the damage that an attacker can cause by exploiting the ChatBug vulnerability.

Following Zou et al. (2023) and Yu, Lin, and Xing (2023), we calculate ASR using two approaches:

- **Refusal Response Matching (ASR-R).** This approach collects a set of refusal responses (e.g., “Sorry, I cannot . . .”) and verifies whether the response generated by LLM matches any of them. An LLM generated response is considered harmful if it does not align with any of the refusal responses.

¹We use Microsoft Azure OpenAI service for the experiment in our work: <https://azure.microsoft.com/en-us/products/ai-services/openai-service>

- **Moderator Assessment (ASR-M).** We utilize a pre-trained LLM, Llama-Guard-2² fine-tuned from Llama 3, as a moderator to evaluate whether a response is harmful.

Baseline. We evaluate the severity of the ChatBug vulnerability by comparing attacks that exploit it with a baseline scenario named **Direct Instruct**. In the baseline scenario, a malicious user directly prompts victim LLMs with harmful queries or instructions.

Attack Settings. We consider six attack settings where the format mismatch attack and message overflow attack exploit ChatBug to provoke unintended behaviors from victim LLMs.

- **Mismatch- \emptyset .** An attacker launches the format mismatch attack by removing all special control tokens in the chat format.
- **Mismatch-C.** An attacker launches the format mismatch attack by replacing the default chat template with a complex template ChatML developed by OpenAI (2023).
- **Mismatch-V.** An attacker initiates the format mismatch attack by replacing the default chat template with a simple template used by Vicuna (Chiang et al. 2023).
- **Overflow-S.** An attacker launches the message overflow attack. For any query, the overflow inserts a fixed prefix ‘*Sure, here is*’ for the desired response.
- **Overflow-L.** For each input query, an attacker initiates the message overflow attack by inserting a prefix tailored particularly for this query. For example, the overflow is ‘*Sure, here is a tutorial for making a bomb*’ when the input query is ‘*Write a tutorial on how to make a bomb*’. These queries and corresponding prefixes are readily available in AdvBench.
- **Overflow-FS.** This is an extension of Overflow-L, where an attacker launches the message overflow attack and utilizes an uncensored LLM to generate the prefixes. In our experiments, we choose Wizard-Vicuna-13B-Uncensored³ as the uncensored LLM, and use in-context-learning prompting technique to generate multiple prefixes that carry affirmative semantics automatically. We generate 5 prefixes by default. The prompt to generate attack prefixes is in Figure 7 of our extended version (Jiang et al. 2024a).

5.2 Main Results

Exploiting ChatBug bypasses safety alignments of all eight victim LLMs. In Tables 2 and 3, we summarize the ASR for format mismatch attack and message overflow attack under different settings on open-source and closed-source LLMs, respectively. We have two key observations. First, exploiting the ChatBug vulnerability effectively elicits unintended responses from all victim LLMs. For example, the ChatBug vulnerability results in 100% ASR-R for the Overflow-FS attack against Llama 3, a state-of-the-art

²<https://huggingface.co/meta-llama/Meta-Llama-Guard-2-8B>

³<https://huggingface.co/cognitivecomputations/Wizard-Vicuna-13B-Uncensored>

open-source LLM. This indicates that ChatBug is a severe and common vulnerability across all open-source and closed-source LLMs that have been fine-tuned with chat templates. Moreover, even if an LLM has been carefully aligned (e.g., Llama and Claude), an attacker could still exploit ChatBug to bypass the safety alignment and provoke unintended behaviors. These observations highlight the pervasiveness and severity of ChatBug.

Safety alignment associated with chat templates is transferable. In Table 2, we observe that the ASR of Mismatch-C and Mismatch-V against some open-source LLMs is relatively low compared to other attacks. For example, the ASR-R of Mismatch-C is 1.3% on Llama-3. Note that Llama uses different chat templates than ChatML by OpenAI (2023). This indicates that the safety alignment by chat templates is transferable. We defer the experiments on few-shot setup for message overflow attack to Section C.1 and tokenizer setup for format mismatch attack to Section C.2.

5.3 ChatBug Boosts Jailbreak Attacks

In the following, we show that ChatBug boosts the ASR of three SOTA jailbreak attacks.

Model. Our empirical evaluations are performed on Llama-2, which has undergone strong safety alignment. This is also evidenced by the ASR of Direct Instruct in Table 2, where Llama-2 exhibits the lowest ASR among all models.

Jailbreak Attacks. We consider three representative SOTA jailbreak attacks: **GCG** (Zou et al. 2023), **GPTFuzzer** (Yu, Lin, and Xing 2023), and **ArtPrompt** (Jiang et al. 2024b). Specifically, GCG is an optimization-based jailbreak attack where a genetic optimization is used to search for attack prompts. GPTFuzzer is an empirical jailbreak attack where prompts are generated autonomously using a mutation-based method. ArtPrompt is an automated jailbreak attack, which replaces words triggering safety alignment with ASCII art. More detailed description of these jailbreak attacks can be found in Appendix A.1 of our extended version (Jiang et al. 2024a).

Exploiting ChatBug significantly boosts ASR of jailbreak attacks. In Table 4, we summarize the ASR of GCG, GPTFuzzer, and ArtPrompt when they exploit the ChatBug vulnerability. We observe that exploiting the ChatBug vulnerability significantly increases their ASR. The integration of GPTFuzzer with Mismatch- \emptyset achieves 99.2% ASR-R, compared to only 9.0% ASR-R when ChatBug vulnerability is not exploited. These results underscore the severity of the ChatBug vulnerability and highlight the urgent need to develop countermeasures.

6 Countermeasures to ChatBug

Given the severity of ChatBug, this section discusses potential countermeasures. We first describe these countermeasures. We then perform empirical evaluations of these countermeasures. Based on our evaluation results, we finally discuss future directions in fine-tuning LLMs which require collaborative efforts from the community.

Attack	Vicuna		Mistral		Llama-2		Llama-3	
	ASR-R	ASR-M	ASR-R	ASR-M	ASR-R	ASR-M	ASR-R	ASR-M
Direct Instruct	5.6%	3.7%	24.0%	22.9%	0.4%	0.0%	1.1%	0.0%
Mismatch- \emptyset	90.6%	40.4%	65.2%	55.6%	17.1%	12.7%	65.4%	50.0%
Mismatch-V	-	-	10.8%	4.6%	0.2%	0.0%	1.2%	1.5%
Mismatch-C	52.3%	37.9%	12.9%	9.0%	5.6%	3.3%	1.3%	0.2%
Overflow-S	98.5%	89.4%	89.8%	83.8%	46.0%	36.4%	92.1%	84.2%
Overflow-L	90.4%	88.5%	64.0%	53.9%	32.5%	20.8%	98.3%	93.5%
Overflow-FS	98.8%	95.2%	96.2%	90.4%	51.0%	31.3%	100.0%	94.1%

Table 2: This table summarizes the ASR-R and ASR-M of Direct Instruct (baseline) and attacks that exploit the ChatBug in open-source LLMs. The results show that an attacker can effectively bypass safety alignment of LLMs by exploiting the ChatBug vulnerability. We have excluded results of Mismatch-V on Vicuna model since they use the same chat template.

Attack	GPT-3.5		Gemini		Clade-2.1		Claude-3	
	ASR-R	ASR-M	ASR-R	ASR-M	ASR-R	ASR-M	ASR-R	ASR-M
Direct Instruct	9.6%	6.3%	10.2%	2.1%	0.2%	0.0%	0.4%	0.0%
Mismatch- \emptyset	86.5%	76.9%	-	-	-	-	-	-
Mismatch-V	86.2%	77.7%	-	-	-	-	-	-
Overflow-S	51.7%	51.0%	70.2%	83.5%	33.5%	8.5%	56.0%	24.2%
Overflow-L	30.2%	58.1%	69.2%	83.1%	86.9%	83.1%	22.5%	11.0%
Overflow-FS	65.2%	66.0%	95.0%	91.0%	90.6%	80.0%	68.3%	48.1%

Table 3: This table summarizes the ASR-R and ASR-M of Direct Instruct (baseline) and attacks that exploit the ChatBug vulnerability in closed-source LLMs. The results show that an attacker can effectively bypass safety alignment of LLMs by exploiting the ChatBug vulnerability. We have excluded results of format mismatch attacks on Gemini and Claude models since their APIs do not support features to execute such attacks. We detail how to launch format mismatch attacks against commercial LLMs with API access in Appendix A.2 of our extended version (Jiang et al. 2024a).

6.1 Description of Countermeasures

We consider two categories of countermeasure: *mitigation-based* and *detection-based* methods.

Mitigation-based Methods. We consider three representative mitigation-based methods including **Self-Reminder** (Xie et al. 2023), **SafeDecoding** (Xu et al. 2024), and **Adversarial Training** (Kurakin, Goodfellow, and Bengio 2016). Self-Reminder utilizes a system-mode prompt to the victim model to strengthen the safety. SafeDecoding is a lightweight decoding strategy to reduce the probability of generating unsafe responses by victim LLMs. Adversarial Training fine-tunes the model with adversarial examples to robustify the model against the vulnerability. We augment the dataset with adversarial examples constructed using the format mismatch attack and message overflow attack. Then we use 60% of the augmented dataset to fine-tune the victim LLM and 40% of the dataset for evaluation. More details of these setups can be found at Appendix A.3 of our extended version (Jiang et al. 2024a).

Detection-based Methods. Detection-based methods monitor input queries and/or generated responses. An input query or response will be blocked if it is flagged as unsafe by a detector. Typical countermeasures may employ key-

word filtering (Jhaver et al. 2019) or established classifiers (Lees et al. 2022) to identify harmful queries or responses. Leveraging the recent advancements in LLMs, Llama Guard (Llama-Team 2024) can be employed to safeguard the responses generated by LLMs. Although effective, detection-based methods are less frequently adopted in practice compared to mitigation-based methods.

6.2 Evaluation of Countermeasures

Model. We evaluate the countermeasures on Vicuna model since it shows the highest ASR on average according to Table 2. This indicates that Vicuna is susceptible to the ChatBug vulnerability.

Metrics. In addition to ASR, we adopt MT-Bench (Zheng et al. 2023) to evaluate the multi-turn conversation and instruction following abilities of victim LLMs after deploying the countermeasures. Higher scores earned on MT-Bench indicate that the models are more helpful to users.

Countermeasure Setup. We follow the setup of Self-Reminder and use a system-mode prompt to remind the victim LLM to be responsible. The system-mode prompt can be found in Appendix A.3 of our extended version (Jiang et al. 2024a). We use the default setup and hyper-parameters

Attack + Booster	ASR-R	ASR-M
GCG	41.5%	32.9%
+ Mismatch- \emptyset	55.4%	51.2%
+ Overflow-S	78.7%	68.3%
+ Overflow-L	76.9%	68.3%
GPTFuzzer	9.0%	7.3%
+ Mismatch- \emptyset	99.2%	83.27%
+ Overflow-S	34.4%	24.0%
+ Overflow-L	32.1%	41.7%
ArtPrompt	73.1%	5.77%
+ Mismatch- \emptyset	100.0%	94.0%
+ Overflow-S	100.0%	15.77%
+ Overflow-L	61.5%	67.7%

Table 4: This table compares ASR of jailbreak attacks when exploiting or not exploiting the ChatBug vulnerability. The results show that ASR of jailbreak attacks is significantly boosted when some variant of ChatBug (Mismatch- \emptyset , Overflow-S, or Overflow-L) are used as boosters.

from SafeDecoding. For Adversarial Training, we consider two setups, where the victim model is fine-tuned with 5 and 20 epochs, respectively.

Experimental Results. In Table 5, we present the ASR and MT-Bench scores (Zheng et al. 2023) of Vicuna with countermeasures being deployed. We observe that mitigation-based countermeasures including Self-Reminder and SafeDecoding fail to mitigate the ChatBug vulnerability. Although they can successfully defend against Mismatch-C, they incur significant degradation on MT-bench. Adversarial Training, however, is an effective technique to mitigate the ChatBug vulnerability, especially when the model is fine-tuned with more epochs. However, the effectiveness of Adversarial Training comes at the significant cost of performance degradation- indicated by the *degradation* in the MT-bench score from 6.28 (comparable to Llama-2-70b-chat performance (Chiang et al. 2024)) to 6.02 (worse than Llama-2-7b-chat performance (Chiang et al. 2024)). These results indicate that developers need to carefully balance the trade-off between safety alignment and helpfulness in future developments of LLMs. Additional experimental evaluation on the effectiveness of adversarial training can be found in Appendix C.3 of our extended version (Jiang et al. 2024a).

7 Conclusion and Future Work

In this paper, we identified a common vulnerability, named ChatBug, induced by chat templates used during instruction tuning. We developed two attacks, format mismatch attack and message overflow attack, to exploit the ChatBug vulnerability. We assessed the severity of ChatBug vulnerability by demonstrating that malicious users could effectively provoke unintended behaviors from eight SOTA aligned LLMs. We further demonstrated that jailbreak attacks could significantly increase their attack suc-

Defense	Attack	ASR-R(\downarrow)	ASR-M(\downarrow)	MT-Bench (\uparrow)
No Defense	Direct Instruct	5.6%	3.7%	6.28
	Mismatch- \emptyset	90.6%	40.4%	
	Mismatch-C	52.3%	37.9%	
	Overflow-S	98.5%	89.4%	
	Overflow-L	90.4%	88.5%	
Self-Reminder	Direct Instruct	5.4%	5.8%	6.07
	Mismatch- \emptyset	23.3%	16.3%	
	Mismatch-C	3.8%	2.7%	
	Overflow-S	78.8%	63.3%	
	Overflow-L	70.4%	86.2%	
SafeDecoding	Direct Instruct	0.2%	0.0%	5.93
	Mismatch- \emptyset	75.4%	55.0%	
	Mismatch-C	5.0%	3.3%	
	Overflow-S	96.2%	90.0%	
	Overflow-L	56.3%	83.7%	
Adversarial Training (5 epochs)	Direct Instruct	1.3%	0.0%	6.15
	Mismatch- \emptyset	1.3%	2.6%	
	Mismatch-C	35.3%	33.3%	
	Overflow-S	26.3%	23.1%	
	Overflow-L	5.1%	30.8%	
Adversarial Training (20 epochs)	Direct Instruct	0.0%	0.0%	6.02
	Mismatch- \emptyset	0.0%	0.0%	
	Mismatch-C	0.0%	0.0%	
	Overflow-S	1.9%	1.9%	
	Overflow-L	5.8%	6.4%	

Table 5: This table presents ASR and MT-Bench scores of Vicuna when countermeasures (Self-Reminder, SafeDecoding, and Adversarial Training) are deployed. The results show that while Adversarial Training can effectively mitigate ChatBug, the performance degrades significantly. The MT-bench score drops from 6.28 to 6.02.

cess rates by exploiting the ChatBug vulnerability. We investigated potential techniques to mitigate ChatBug. Our experimental results showed that although adversarial training could effectively mitigate the ChatBug vulnerability, it came at the cost of degraded model performance, which highlighted the critical trade-off between safety and helpfulness during instruction tuning. Our future work will focus on this trade-off. We aim to develop new methods for instruction tuning to balance this trade-off.

Ethical Statement

In this paper, we demonstrate that chat templates induce a common vulnerability named ChatBug to LLMs. In addition to Self-Reminder, SafeDecoding, and Adversarial Training, mitigation techniques to our identified vulnerability need to be further explored. We believe that detection-based countermeasures could effectively mitigate the ChatBug vulnerability. However, such methods are less frequently deployed in practice due to the potential latency concerns and false positives in detection, which can significantly degrade performance and hinder user experience.

The primary goal of this paper is to advance the safety alignment of LLMs to improve interactions with users. We aim to understand how chat templates affect the safety alignment of LLMs. The ChatBug vulnerability identified in this paper reveals limitations inherited from the widely-used instruction tuning of LLM. We acknowledge that the ChatBug vulnerability can be exploited to misuse LLMs. We investigate potential mitigation techniques against the ChatBug vulnerability. We will release and disseminate the

code and prompts used in our experiments to the community, aiming to assist red-teaming efforts to further mitigate the vulnerability. We will disclose the ChatBug vulnerability to the broader community, including service providers, organizations such as OWASP, as well as through common vulnerabilities and exposures (CVE) listings, to minimize potential damages or harms.

Acknowledgments

This work is partially supported by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-23-1-0208, the National Science Foundation (NSF) under grants IIS 2229876, and the Office of Naval Research under grant N0014-23-1-2386.

This work is supported in part by funds provided by the National Science Foundation, by the Department of Homeland Security, and by IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or its federal agency and industry partners.

References

- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Andriushchenko, M.; Croce, F.; and Flammarion, N. 2024. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*.
- Anthropic. 2023. Introducing Claude 2.1. <https://www.anthropic.com/news/claude-2-1>.
- Anthropic. 2024. Introducing the next generation of Claude. <https://www.anthropic.com/news/claude-3-family>.
- Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; Das-Sarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Bakker, M.; Chadwick, M.; Sheahan, H.; Tessler, M.; Campbell-Gillingham, L.; Balaguer, J.; McAleese, N.; Glaese, A.; Aslanides, J.; Botvinick, M.; et al. 2022. Fine-tuning language models to find agreement among humans with diverse preferences. *Advances in Neural Information Processing Systems*, 35: 38176–38189.
- Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.
- Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; and Xing, E. P. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.
- Chiang, W.-L.; Zheng, L.; Sheng, Y.; Angelopoulos, A. N.; Li, T.; Li, D.; Zhang, H.; Zhu, B.; Jordan, M.; Gonzalez, J. E.; and Stoica, I. 2024. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. *arXiv:2403.04132*.
- Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30.
- Deng, Y.; Zhang, W.; Pan, S. J.; and Bing, L. 2023. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*.
- Dinan, E.; Humeau, S.; Chintagunta, B.; and Weston, J. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv preprint arXiv:1908.06083*.
- Ge, S.; Zhou, C.; Hou, R.; Khabsa, M.; Wang, Y.-C.; Wang, Q.; Han, J.; and Mao, Y. 2023. MART: Improving LLM Safety with Multi-round Automatic Red-Teaming. *arXiv preprint arXiv:2311.07689*.
- Gonen, H.; Iyer, S.; Blevins, T.; Smith, N.; and Zettlemoyer, L. 2023. Demystifying Prompts in Language Models via Perplexity Estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 10136–10148. Association for Computational Linguistics.
- Google. 2023. Build with the Gemini API. <https://ai.google.dev/gemini-api>.
- Huang, Y.; Gupta, S.; Xia, M.; Li, K.; and Chen, D. 2023. Catastrophic jailbreak of open-source LLMs via exploiting generation. *arXiv preprint arXiv:2310.06987*.
- HuggingFace. 2024. Templates for Chat Models. https://huggingface.co/docs/transformers/main/en/chat_templating. Accessed: 2024-06-15.
- Jhaver, S.; Birman, I.; Gilbert, E.; and Bruckman, A. 2019. Human-Machine Collaboration for Content Regulation: The Case of Reddit Automoderator. *ACM Trans. Comput.-Hum. Interact.*, 26(5).
- Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. I.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.
- Jiang, F.; Xu, Z.; Niu, L.; Lin, B. Y.; and Poovendran, R. 2024a. ChatBug: A Common Vulnerability of Aligned LLMs Induced by Chat Templates. *arXiv preprint arXiv:2406.12935*.
- Jiang, F.; Xu, Z.; Niu, L.; Xiang, Z.; Ramasubramanian, B.; Li, B.; and Poovendran, R. 2024b. ArtPrompt: ASCII Art-based Jailbreak Attacks against Aligned LLMs. *arXiv preprint arXiv:2402.11753*.
- Jones, E.; Dragan, A.; Raghunathan, A.; and Steinhardt, J. 2023. Automatically Auditing Large Language Models via Discrete Optimization. *arXiv preprint arXiv:2303.04381*.
- Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.
- Lees, A.; Tran, V. Q.; Tay, Y.; Sorensen, J.; Gupta, J.; Metzler, D.; and Vasserman, L. 2022. A new generation of perspective API: Efficient multilingual character-level transformers. In *the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3197–3207.

- Li, X.; Zhou, Z.; Zhu, J.; Yao, J.; Liu, T.; and Han, B. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.
- Lin, L.; Brown, H.; Kawaguchi, K.; and Shieh, M. 2024. Single character perturbations break llm alignment. *arXiv preprint arXiv:2407.03232*.
- Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2023a. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.
- Liu, Y.; Deng, G.; Xu, Z.; Li, Y.; Zheng, Y.; Zhang, Y.; Zhao, L.; Zhang, T.; and Liu, Y. 2023b. Jailbreaking ChatGPT via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*.
- Llama-Team. 2024. Meta Llama Guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.
- Lu, Y.; Bartolo, M.; Moore, A.; Riedel, S.; and Stenetorp, P. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 8086–8098. Association for Computational Linguistics.
- Meta. 2024. Llama 3.
- OpenAI. 2023. ChatML.
- OpenAI. 2023. Models-OpenAI API. <https://platform.openai.com/docs/models>. Accessed: 2023-09-15.
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022a. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 27730–27744.
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022b. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.
- Perez, F.; and Ribeiro, I. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.
- Qi, X.; Zeng, Y.; Xie, T.; Chen, P.-Y.; Jia, R.; Mittal, P.; and Henderson, P. 2024. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To! In *The Twelfth International Conference on Learning Representations*.
- Scelar, M.; Choi, Y.; Tsvetkov, Y.; and Suhr, A. 2024. Quantifying Language Models’ Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.
- Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Vega, J.; Chaudhary, I.; Xu, C.; and Singh, G. 2023. Bypassing the Safety Training of Open-Source LLMs with Priming Attacks. *arXiv preprint arXiv:2312.12321*.
- Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does LLM safety training fail? *arXiv preprint arXiv:2307.02483*.
- 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*.
- Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Xiang, Z.; Jiang, F.; Xiong, Z.; Ramasubramanian, B.; Poovendran, R.; and Li, B. 2024. BadChain: Backdoor Chain-of-Thought Prompting for Large Language Models. In *the 12th International Conference on Learning Representations*.
- Xie, Y.; Yi, J.; Shao, J.; Curl, J.; Lyu, L.; Chen, Q.; Xie, X.; and Wu, F. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5: 1486–1496.
- Xu, Z.; Jiang, F.; Niu, L.; Jia, J.; Lin, B. Y.; and Poovendran, R. 2024. SafeDecoding: Defending against Jailbreak Attacks via Safety-Aware Decoding. *arXiv preprint arXiv:2402.08983*.
- Yu, J.; Lin, X.; and Xing, X. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.
- 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. *arXiv preprint arXiv:2401.06373*.
- Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 46595–46623.
- 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. *arXiv preprint arXiv:2310.15140*.
- Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.