Reinforcement Learning-Driven LLM Agent for Automated Attacks on LLMs

Xiangwen Wang[1], Jie Peng[1], Kaidi Xu[2], Huaxiu Yao[3], Tianlong Chen[3]
[1]University of Science and Technology of China
[2]Drexel University
[3]University of North Carolina at Chapel Hill
{wangxiangwen,pengjieb}@mail.ustc.edu.cn, kx46@drexel.edu
{tianlong, huaxiu}@cs.unc.edu

Abstract

Recently, there has been a growing focus on conducting attacks on large language models (LLMs) to assess LLMs' safety. Yet, existing attack methods face challenges, including the need to access model weights or merely ensuring LLMs output harmful information without controlling the specific content of their output. Exactly control of the LLM output can produce more inconspicuous attacks which could reveal a new page for LLM security. To achieve this, we propose RLTA: the **R**einforcement **L**earning **T**argeted **A**ttack, a framework that is designed for attacking language models (LLMs) and is adaptable to black box (weight inaccessible) scenarios. It is capable of automatically generating malicious prompts that trigger target LLMs to produce specific outputs. We demonstrate RLTA in two different scenarios: LLM trojan detection and jailbreaking. The comprehensive experimental results show the potential of RLTA in enhancing the security measures surrounding contemporary LLMs.

1 Introduction

Recent LLMs have demonstrated remarkable capabilities in a wide range of applications (Achiam et al., 2023; Touvron et al., 2023). However, LLMs are susceptible to various security vulnerabilities, including adversarial attacks and unintended behaviors (Bommasani et al., 2021; Bender et al., 2021; Gehman et al., 2020; Weidinger et al., 2021), focus attention of pioneers in LLM attack. Existing attack methods can induce models to make errors or generate harmful content (Zhang et al., 2020; Jia and Liang, 2017; Guo et al., 2021; Zou et al., 2023; Shen et al., 2023; Chao et al., 2023; Wei et al., 2023).

However, some existing methods rely on handcrafted prompts produced by experts which are domain-specific and often labor-intensive (walkerspider, 2022; Wei et al., 2023), and many of these handcrafted prompts speedily failed in subsequently released models like ChatGPT-4 (Achiam et al., 2023), also lacks control on LLM specific output. Methods like Guo et al., 2021 and Zou et al., 2023 can force models to output specific content but require the assessment of their weights.

To address these challenges, we propose the novel **R**einforcement **L**earning **T**argeted **A**ttack (RLTA) framework, leveraging reinforcement learning (RL) to train a language model as the agent that controls the target LLM into generating desired content. Given the specific output that the target model is intended to produce, the LM agent creates a corresponding prompt, which is then utilized as the input of the target LLM. The effectiveness of the prompt is assessed based on the response it elicits from the target model, and this feedback is used to optimize the agent model through Proximal Policy Optimization (PPO) (Schulman et al., 2017). After training, the LM agent can generate the prompt that can induce the target LLM to output the target content. By leveraging the generalizability of language models, the trained LM agent is able to generate corresponding prompts for unseen target outputs. Additionally, leveraging RL, our approach naturally works on black box LLMs of which the gradient information is inaccessible, which broadens its applicability. Furthermore, the RLTA exactly controls the target LLM output, introducing a more secretive LLM attack which paves the path for the next era of LLM attack.

In summary, our main contributions are as follows: (1) we introduce a novel framework that utilizes reinforcement learning to train an agent model that automatically generates malicious prompts, which can be used for black-box settings. (2) Our approach achieves high Attack Success Rates (ASR) and demonstrates precise control over the outputs of target language models, ensuring that the generated content closely aligns with predefined harmful objectives. (3) The versatility of our

Proceedings of the Fifth Workshop on Privacy in Natural Language Processing, pages 170–177

framework allows for broad generalization across multiple tasks. (4) We apply our method to the unexplored area of trojan detection through reverse engineering, revealing its potential to uncover and understand hidden malicious configurations within language models.

2 Related Works

Reinforcement Learning for LLMs. Recent research has explored various aspects of using LLMs as agents in RL environments where natural language is used as the state or action space of the agent (Alabdulkarim et al., 2021; Carta et al., 2023; Shinn et al., 2024; Zhang et al., 2024; Dognin et al., 2021). Reinforcement Learning from Human Feedback (RLHF) is a typical application of leveraging RL to fine-tune LLMs (Christiano et al., 2017; MacGlashan et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Glaese et al., 2022), where the reward score provided by the reward model is utilized to enhance the agent's performance using policy gradient algorithms(Schulman et al., 2017). Moreover, Perez et al., 2022 leveraged RL to train a language model to red-team another language model, excepting the target model to generate harmful content indiscriminately. Our strategy aims to exert precise control of the output content over the target model's responses using RL.

Jailbreaking LLMs. Aligned language models (Achiam et al., 2023; Ouyang et al., 2022; Touvron et al., 2023) are vulnerable to jailbreaking prompts designed to manipulate responses in harmful or biased ways. Hand-crafted methods like DAN(walkerspider, 2022) rely on manual creation but are domain-specific and labor-intensive(walkerspider, 2022; Wei et al., 2023; Gehman et al., 2020). Optimization-based methods, which append adversarial suffixes to prompts and require model gradient information, are detectable through perplexity-based checks (Ebrahimi et al., 2017; Jia and Liang, 2017; Wallace et al., 2019; Guo et al., 2021; Zou et al., 2023; Jones et al., 2023). Besides hand-crafted jailbreaking attacks and optimization-based attacks, LLM-based attacks emerged, where another LLM is used to jailbreak the target LLM (Chao et al., 2023; Mehrotra et al., 2023). The PAIR framework, introduced by Chao et al., 2023, involves an attacker LLM iteratively querying the target LLM to refine a candidate jailbreak prompt. Extending this concept, Mehrotra et al., 2023 developed TAP, which enhances the refinement process using tree-of-thought reasoning. Conversely, our method employs reinforcement learning to educate an agent to create jailbreaking prompts through a single forward inference.

3 Method

As shown in Figure 1, our approach employs RL where the agent LM is trained to generate prompts that manipulate the output of a target language model. Given the specific target content, our aim is to use RL to optimize the agent LM so that the output prompts compel the target model to generate the corresponding specific content.

3.1 Preliminary: Reinforcement Learning to Desired Target

RL has proven effective in optimizing LLMs towards a specific goal leveraging reward signals provided by the reward model (Ouyang et al., 2022; Stiennon et al., 2020). Current methods fine-tune the model by the PPO (Schulman et al., 2017) algorithm with the objective function:

$$O(\phi) = \mathbb{E}_{(x,y)\sim D_{\pi_\phi^{RL}}}\left[R(x,y) - \beta \log\left(\frac{\pi_\phi^{RL}(y \mid x)}{\pi^{Init}(y \mid x)}\right)\right], \tag{1}$$

where π_ϕ^{RL}, the LM agent, denotes the learned RL policy with trainable parameter ϕ optimized by the RL training process, π^{Init} indicates the LM agent with parameters frozen before training. The coefficients β regulate the strength of the KL penalty.

3.2 The Reinforcement Learning Targeted Attack Framework

As illustrated in Equation 1, in our framework, x represents the desired harmful output for the target model T. Notably, for the target model T with well-aligned fine-tuning, it will refuse to generate harmful sentence x. The agent model A aims to generate a malicious prompt $y = A(x)$ based on the given x that leads the target model T to produce an output $z = T(y)$, which should align with x.

RLTA Training. We adopt the agent model A as the learning RL policy π_ϕ^{RL}. We initialize π_ϕ^{RL} as a pre-trained language model, denoted as π_{Init}, and freeze the parameters of π_{Init}. The reward function $R(x, y)$ is calculated based on the target model's output $z = T(y)$, where the input of the target model is the malicious prompt generated by π_ϕ^{RL}.

$$R(x,y) = E(x,z) = E(x,T(y)) \tag{2}$$

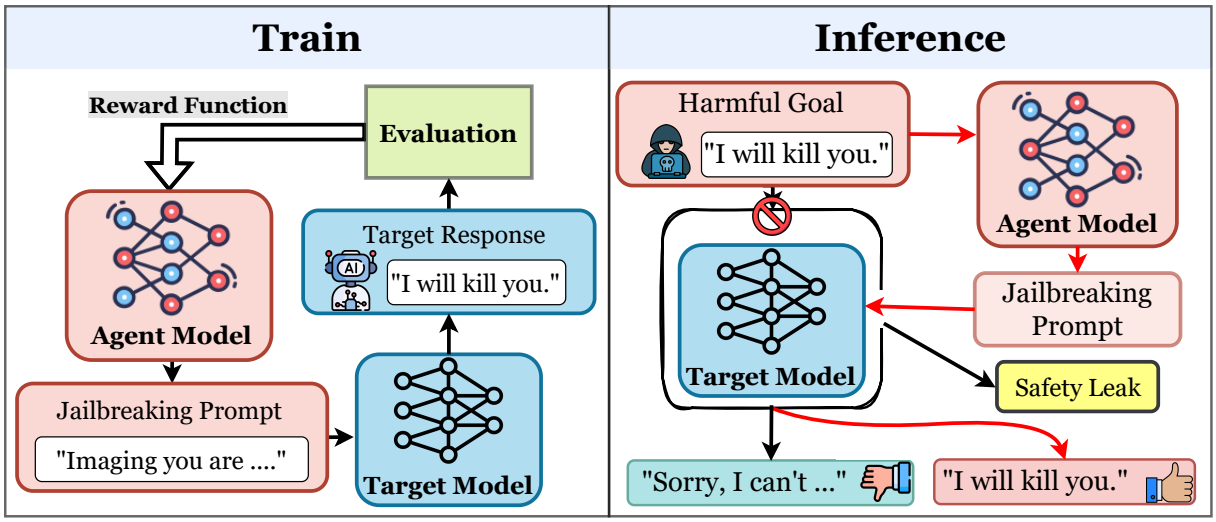

Figure 1: Framework of Reinforcement Learning Targeted Attack (RLTA). **Left** illustrates the training process of RLTA, The objective of the "Agent Model" is to process the "Harmful Goals" and generate "Jailbreaking Prompts". These prompts are then fed into the "Target Model", prompting it to produce outputs that align closely with the "Harmful Goals". Since the gradient cannot backpropagate through "Jailbreaking Prompts", therefore we utilize reinforcement learning to update the "Agent Model". After the attack agent training, during the inference, the **Right** shows that we first input the desired "Harmful Goals" to the agent model. The RLTA then feeds the output from the agent model into the "Target Model" to execute the attack.

The objective function during the training process is calculated as previously done in Equation 1:

Here, D is the set of inputs (desired harmful content) for the agent model, where x is the sampled prompt from D and y is the output generated by π_ϕ^{RL}, which is the malicious prompt.

RLTA Inference. RLTA can generalize effectively to unseen attack goals. When unseen target content is introduced to the trained agent model, the LM agent autonomously generates the corresponding malicious prompt for the target model. This inference process requires only a single forward pass through the agent model. This capability ensures that RLTA can adapt and respond to a variety of scenarios without the need for iterative interaction during the inference phase.

3.3 Frameworks for Different Applications

The method can be applied to several scenarios, including detecting trojans inserted into the target model and jailbreaking the target model to elicit a specific target string.

RLTA for Trojan Detection. In the Trojan Detection scenario, the target model T is inserted into multiple trojans, each defined by a pair of text strings: a trigger and a target: $(S^{(i)}_{\texttt{trigger}}, S^{(i)}_{\texttt{target}})$. The target model will output the target string when the corresponding trigger string is the input:

$$S^{(i)}_{\texttt{target}} = T(S^{(i)}_{\texttt{trigger}}) \tag{3}$$

The agent model's task is to identify $S^{(i)}_{\texttt{trigger}}$ for a given $S^{(i)}_{\texttt{target}}$.

The trigger $y = A(S^{(i)}_{\texttt{target}})$ detected by the agent model is evaluated using two metrics: recall and reverse-engineered attack success rate (REASR). Recall was measured using the BLEU score to compare the predicted triggers with the actual triggers that were initially inserted into the target model. REASR was assessed by the BLEU score between the target strings and the target model's outputs elicited from the predicted triggers. The combination of Recall and REASR is used as reward to train the agent model.

$$\begin{aligned} R(x, y) &= R(S^{(i)}_{\texttt{target}}, y) && (4)\\ &= \alpha \cdot \texttt{Recall} + \beta \cdot \texttt{REASR} && (5)\\ &= \alpha \cdot \texttt{BLEU}(y, S^{(i)}_{\texttt{trigger}}) && (6)\\ &+ \beta \cdot \texttt{BLEU}(T(y), S^{(i)}_{\texttt{target}}) && (7) \end{aligned}$$

RLTA for Jailbreaking. This application involves eliciting a model to produce a specific harmful or misleading string x. The jailbreaking prompt y, generated by the agent model, is evaluated based on the similarity between $T(y)$ and x using the BLEU score.

$$R(x, y) = \texttt{BLEU}(T(y), x) \tag{8}$$

4 Experiments

Datasets. We applied our method to the trojan detection dataset (TDC)(Center for AI Safety, 2023)

Type	Method	Agent	REASR	Recall
Black-box	RLTA(Ours)	Pythia-1.4B	0.94	0.15
		Vicuna-7B	0.38	**0.20**
		Llama-3-8B	0.45	0.14
		Llama-2-7B	0.32	0.20
	PAIR	Vicuna-7B	0.20	0.09
		Llama-3-8B	0.37	0.15
White-box	GCG	–	**0.98**	0.09
	GBDA	–	0.05	0.11
	PEZ	–	0.05	0.11

Table 1: The Reverse-Engineered Attack Success Rate (REASR) and Recall scores for various methods on the Trojan Detection Challenge (TDC) dataset. The methods are categorized into black-box and white-box types. Each method's performance is evaluated using different agent models such as Pythia-1.4B, Vicuna-7B, Llama-3-8B, and Llama-2-7B.

Type	Method	Agent	Target Model		
			Llama-3-7B	Vicuna-7B	Llama-2-7B
Black-box	RLTA(Ours)	Pythia-1.4B	0.32	0.47	0.26
		Llama-3-7B	0.75	0.80	0.76
		Vicuna-7B	0.47	0.37	0.39
		Llama-2-7B	0.33	0.43	0.74
	PAIR	Llama-3-7B	0.24	0.37	0.16
		Vicuna-7B	0.19	0.34	0.22
White-box	GCG	–	**0.89**	**0.93**	**0.87**

Table 2: The Attack Success Rates (ASR) for jailbreaking attacks on the Harmful Strings subset of the AdvBench dataset.The performance of each method is evaluated using different agent models (Pythia-1.4B, Llama-3-7B, Vicuna-7B, and Llama-2-7B) and target models (Llama-3-7B, Vicuna-7B, and Llama-2-7B). The methods are categorized into black-box and white-box types.

and "harmful strings" subset from AdvBench (HS) (Zou et al., 2023), corresponding to trojan detection and jailbreaking application in Section 3.3, respectively. For more details on datasets see Appendix A.

Agent model. We employed several agent models: vanilla Pythia-1.4B (Biderman et al., 2023), Vicuna-7B (Zheng et al., 2024), the newly introduced Llama3-8B-it (Meta, 2024), and Llama2-7B-chat (Touvron et al., 2023).

Target model. For TDC dataset, We followed the setup of Trojan Detection Track of Trojan Detection Challenge 2023 (LLM Edition)(Center for AI Safety, 2023). The challenge provided a target model finetuned from Pythia 1.4B, containing 100 trojans. For HS dataset, we executed attacks on Vicuna-7B (Zheng et al., 2024), Llama3-8B-it (Meta, 2024), and Llama2-7B-chat.

Baselines. Our approach was compared against PAIR (Chao et al., 2023), GBDA (Guo et al., 2021), PEZ (Wen et al., 2024), and GCG attack (Zou et al., 2023). Our method and PAIR were tested in black-box setting, while others in white-box setting.

Metrics. Our evaluation metrics for TDC dataset were recall and reverse-engineered attack success rate (REASR), for HS dataset was attack success rate (ASR), as previously described in Section 3.3.

Results for TDC dataset. The results, displayed in Table 1, include recall and REASR scores for the different methods we tested. Our RLTA method outperformed all other black-box baseline methods and achieved comparable efficiency to the white-box GCG method. Notably, while the ASR scores reached impressively high levels, recall scores remained relatively low across all methods. This discrepancy suggests that the insertion of trojans might make not only the target model sensitive to specific triggers, but other input can elicit targets as well. Moreover, Pythia-1.4B, when used as an agent model, was most effective in detecting trojans within a target model also based on Pythia-1.4B. This highlights the advantage of using agent models similar to the target model. For other agent models, the data reveals that more advanced models can perform the task more efficiently.

Results for HS dataset. The ASR for "harmful strings" dataset is shown in table 2. The results reveal that our method significantly outperformed other black-box approaches in jailbreaking tasks. Similar to the Trojan Detection scenario, ASR scores vary between different agent and target models. The Llama3-8B-it model demonstrated superior performance in generating jailbreaking prompts while Pythia-1.4B model performs worst, indicating that more advanced models have better performance even with different model architectures and pretrained datasets. For the target model, the Vicuna 7B model displayed a higher susceptibility to our RLTA jailbreaking prompts compared to Llama3-8B and Llama2-7B.

5 Conclusion

In this paper, we have introduced a novel reinforcement learning-based framework, `RLTA`, for the targeted attack of LLMs. Our approach leverages the capabilities of reinforcement learning to train an LLM agent that can autonomously generate malicious prompts to manipulate the output of target LLMs in black-box settings. The effectiveness of our method is demonstrated through extensive experiments involving different scenarios, including trojan detection and jailbreaking to induce specific harmful outputs.

6 Limitation.

Our experiments were mainly conducted on models up to 8B and did not include testing on larger open-source models or closed-source models. The effectiveness of our RLTA framework on these larger and potentially more complex models remains unverified, which may limit the generalizability of our results. Future studies should aim to apply and validate our method across a broader spectrum of LLMs to fully understand its potential and limitations in real-world scenarios.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Amal Alabdulkarim, Winston Li, Lara J Martin, and Mark O Riedl. 2021. Goal-directed story generation: Augmenting generative language models with reinforcement learning. *arXiv preprint arXiv:2112.08593*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR.

Center for AI Safety. 2023. The trojan detection challenge 2023 (llm edition).

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Pierre L Dognin, Inkit Padhi, Igor Melnyk, and Payel Das. 2021. Regen: Reinforcement learning for text and knowledge base generation using pretrained language models. *arXiv preprint arXiv:2108.12472*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. *arXiv preprint arXiv:2104.13733*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Erik Jones, Anca Dragan, Aditi Raghunathan, and Jacob Steinhardt. 2023. Automatically auditing large language models via discrete optimization. In *International Conference on Machine Learning*, pages 15307–15329. PMLR.

James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *International conference on machine learning*, pages 2285–2294. PMLR.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*.

Meta. 2024. meta-llama/meta-llama-3-8b-instruct.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

walkerspider. 2022. Dan is my new friend.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? *Preprint*, arXiv:2307.02483.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. 2024. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Datasets

A.1 Trojan detection application

Trojan Detection Track of Trojan Detection Challenge 2023 (LLM Edition)(Center for AI Safety, 2023) provided a target model finetuned from Pythia 1.4B, which was inserted with 100 trojans during the finetuning process. Each trojan is defined by a (trigger, target) pair, where the target strings are harmful content. The target model will output the target string when the corresponding trigger string is the input.

We utilized 80 of these trojans, including both trigger strings and target strings, as the training set. The target strings serve as predefined harmful outputs. During training, we input the target strings into our agent model and let it predict the corresponding triggers. The predicted triggers are evaluated using the Reverse-Engineered Attack Success Rate (REASR) and Recall metrics, and these evaluations are used as rewards to train the agent model.

The remaining 20 trojans were used as the test set. In this phase, the agent model predicts the triggers for the unseen targets in the test set. The evaluation of these predicted triggers in the test set constitutes the results of the experiment. Since the predicted triggers can elicit the target model to produce harmful content, this process is viewed as a specialized form of attack.

A.2 Jailbreaking application.

We utilized the "harmful strings" subset from AdvBench (Zou et al., 2023). This subset consists of 500 strings that reflect harmful or toxic behavior. The goal for the attacker is to discover specific inputs that can prompt the model to generate these exact harmful strings.

We randomly split the dataset in 8:2 for training set and test set. During the training phase, our agent model is tasked with discovering inputs that can lead the target model to produce the predefined harmful outputs. These generated inputs are then fed into the target model, and the target model's outputs are compared to the harmful strings. This evaluation process serves as the reward for training the agent model. Unlike the Trojan detection application, there are no ground truth inputs for the target model in this case. Therefore, the inputs discovered by the agent model are evaluated based on Attack Success Rate (ASR).

For the test phase, the trained agent model generates inputs for the unseen harmful targets in the test set. The effectiveness of these inputs is again evaluated using ASR, and this evaluation constitutes the results of the experiment like Trojan detection dataset.

B Training Configurations

B.1 Training Details

The agent model was trained using the PPO algorithm with the following hyperparameters:

- Learning Rate: 1e-6
- KL penalty coefficient: 0.03
- Batch Size: 8
- Number of Epochs: 30
- Clip Range: 0.3

B.2 Computational Resources

For the TDC dataset, training was conducted on an NVIDIA RTX 3090 GPU with 24GB of RAM, and the training duration for the agent model was approximately 15 hours. For the harmful strings dataset, training was conducted on an NVIDIA A6000 GPU with 48GB of RAM, and the training duration for the agent model was approximately 96 hours.

C Relationship Between Attack Methods and Privacy

The primary focus of our research is on developing and evaluating reinforcement learning-based attack methods to expose vulnerabilities in large language models (LLMs). These methods, specifically Trojan detection and jailbreaking, aim to manipulate LLMs to produce harmful outputs. While these attacks are primarily designed to assess and improve the security of LLMs, they have significant privacy implications that must be considered. For instance, triggering hidden behaviors might lead to the unintentional disclosure of private data that the model has been exposed to during training. Jailbreaking prompts can also potentially manipulate LLMs to reveal private or sensitive information that should be protected. While the attack methods proposed in this paper are crucial for enhancing the security and robustness of LLMs, it is imperative to recognize and address the privacy implications associated with these techniques.